UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS - UFR Sciences

ÉCOLE DOCTORALE EDSFA
SCIENCES FONDAMENTALES ET APPLIQUÉES

T H È S E

pour obtenir le titre de

Docteur en Sciences

de l'Université de Nice - Sophia Antipolis
Mention : MATHEMATIQUES

Présentée et soutenue par

Joseph CHARLES

Amélioration des performances de méthodes Galerkin discontinues d'ordre élevé pour la résolution numérique des équations de Maxwell instationnaires sur des maillages simplexes

Thèse dirigée par Stéphane LANTERI

préparée à l'INRIA Sophia Antipolis, Projet NACHOS

soutenue le 26 Avril 2012

Jury :

Président :	Boniface NKONGA	-	Professeur à l'Université de Nice - Sophia Antipolis
Directeur :	Stéphane LANTERI	-	Directeur de Recherche à l'INRIA Sophia Antipolis - Méditerranée
Rapporteurs :	Luc GIRAUD	-	Directeur de Recherche à l'INRIA (HiePACS)
	Damien TROMEUR-DERVOUT	-	Professeur à l'Université de Lyon 1
Examinateurs :	Muriel SESQUES	-	Ingénieur de Recherche au CEA/CESTA

A la mémoire de mon grand père Lionel, ma grande tante Lolotte, Jean Claude, Pierre et Claire

Pour mon grand père Lionel

Remerciements

Depuis le premier jour de mon arrivée sur la Côté d'Azur, un peu plus de trois années se sont écoulées est le moment est venu pour moi d'adresser mes remerciements.

Mes premiers remerciements vont évidemment à mon directeur de thèse, Stéphane Lantéri, pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail doctoral, pour ses nombreux conseils scientifiques, et pour toutes les heures et qu'il a consacrées à diriger cette recherche. Je lui suis aussi particulièrement reconnaissant pour m'avoir proposé un sujet de thèse en relation avec le calcul haute performance, voie dans laquelle je me dirige à présent. Au delà de ma gratitude pour m'avoir transmis un peu de sa connaissance scientifique et intellectuelle pendant trois ans, j'aimerais surtout lui dire à quel point j'ai apprécié sa grande disponibilité et son soutien dans des périodes particulièrement dures de ma vie, notamment lorsque mon grand père Lionel est décédé et lorsque peu de temps après, mon père fut touché d'une rupture d'anévrisme. Dans ces moments douloureux, j'ai été particulièrement sensible à ta bienveillance et à tes qualités humaines d'écoute et de compréhension. C'est une image forte que je garderais de notre collaboration et pour cela je te remercie profondément.

C'est avec un grand plaisir que je remercie également tous les autres membres, anciens et actuels, de l'équipe projet NACHOS : Nathalie Glinsky, Montserrat Argente, Loula Fezoui, Stéphane Descombes, Victoria Dolean, Claire Scheid, Liang Li, Nora Aissiouene, Chrisitian Konrad, Clément Durochat, Mohamed El Bouajaji, Tristan Cabel, Fabien Peyrusse, Ludovic Moya et Hassan Fahs pour leur grande sympathie, leur bonne humeur et pour la bonne ambiance générale des pauses cafés et des repas quotidiens. Un grand merci à Montserrat non seulement pour sa gentillesse et son aide dans les papiers administratifs mais aussi pour les nombreuses discussions échangées et son soutien amical dans les moments difficiles. Sur le plan humain, je tiens également à exprimer ma gratitude à Nathalie avec qui j'ai eu beaucoup de plaisir à discuter de sujets très variés. Merci bien sûr d'avoir pris le temps de relire mon manuscrit et de m'avoir conseillé, guidé et aidé comme tu l'as fait. Mes remerciements vont ensuite tout naturellement aux trois personnes qui ont partagé mon bureau et qui ont vécu les péripéties de mon quotidien : Christian, Clément et Tristan grâce auxquels l'intensité du travail doctoral fut beaucoup plus facile à digérer. Merci pour votre joie de vivre, votre écoute, vos attentions et pour m'avoir supporté pendant toutes ces années. J'ai une pensée toute particulière pour Clément qui à son tour se dirige vers la fin de sa troisième année, et qui désormais traverse la tâche ardue de rédaction du manuscrit final. Un grand merci pour ton enthousiasme, ta complicité, ta profonde gentillesse et pour tous les délires passés à tes côtés, bon courage pour la fin ! Je ne pourrais conclure ses remerciements vis à vis de l'équipe sans citer Mohamed avec qui j'ai partagé un nombre incalculable de pauses cafés la nuit au bureau dans le stress et l'épuisement caractéristiques d'une fin de thèse. Merci pour ta disponibilité indéfectible, ton grand sens de l'humour et pour tous ces fous rires échangés pendant ces trois années.

Ce rapport final étant le fruit d'une collaboration, je souhaite témoigner toute ma reconnaissance à Stéphane Lantéri, Nathalie Glinsky, Loula Fezoui, Stéphane Descombes, Tristan Cabel et Julien Diaz qui ont contribué à leur manière au développement et à la finalisation de ce manuscrit.

Ce travail de recherche a été réalisé au sein de l'INRIA Sophia Antipolis dirigé par Monsieur Gérard Giraudon. Je le remercie vivement de m'y avoir accueilli. J'en profite également pour remercier le centre CEA DAM, CESTA de Bordeaux pour son soutien financier au contrat doctoral, ainsi que Muriel SESQUES et ses partenaires pour leur gentillesse et leur écoute attentive à chacune des présentations de mes travaux en cours.

Je remercie Monsieur Luc Giraud et Monsieur Damien Tromeur-Dervout de m'avoir fait le privilège d'être les rapporteurs de cette thèse et d'avoir participé à l'amélioration de mon manuscrit par leurs nombreux commentaires. Merci également à Monsieur Michel Madalena et à Madame Muriel Sesques

pour avoir accepté de faire partie de mon jury de thèse en tant qu'examinatrice. C'est avec un grand plaisir que je remercie enfin, mon ancien professeur à Matméca, Monsieur Boniface Nkonga, pour m'avoir fait l'honneur de présider mon jury de thèse. A tous je les remercie aussi de s'être déplacés de loin pour certains, afin de participer à ma soutenance.

Bien sûr, un énorme merci à tous les potes, les amis rencontrés pendant cette période de vie intense, que ce soit dans le cadre du travail ou en dehors, qui m'ont aidé d'une manière ou d'une autre à vivre trois années riches en émotions et inoubliables : Julien C., Alina, Ben, Manu, Sylvain, Rania, Jean, Anca, Federica, Bogdan, Sapna, Meriem, Ana, Mandar, Fabien, Giovani, Zara, Romain, Germain, Nicolas T., Marcela, Krissy, Kristina, Ruta, Guillaume, Magda, Marc, Nicolas C., Hubert, Antoine. Merci aussi au groupe de jonglage INRIA-Amadeus et particulièrement à Johanna, Manu, Laurent, Fred, Julien, Xavier, Jean Guillaume et Fabrice pour la bouffée d'oxygène que vous m'avez apportée et pour les moments de détente à peaufiner les gestes en bord de mer ou dans la forêt. Une grosse pensée également à mon ancien club de natation, le Cercle des Nageurs d'Antibes, dans lequel j'adorais me retrouver pour enchaîner les longueurs sous le soleil, été comme hiver, ce qui me manque cruellement à présent. J'adresse enfin un énorme remerciement aux potes des Adrechs', cette fine équipe de bons vivants partageant un sens aigu de la fête doublé d'une convivialité et du plaisir d'être ensemble.

Sur ces trois années de vie Antiboise, trois collocataires se sont succédés et ont du supporter courageusement mon sale caractère et mon côté un peu maniaque. J'aimerais leur témoigner ma profonde reconnaissance pour tout ce qu'ils ont pu endurer. Toutefois, la médaille du mérite revient à mon vieil acolyte Ju pour sa patience mise à rude épreuve par mes humeurs durant la fin de sa thèse et pour avoir tant de fois tenté en vain de résister aux sollicitations d'amis peu scrupuleux. Pour ta générosité sans limite, ton écoute, ton inconditionnel soutien et pour toutes ces belles et longues années d'amitié parfumées d'ivresse et ponctuées d'anecdotes en tous genres , je t'adresse un très grand merci ! Bien qu'il m'a fallut faire face à ton humour franchement médiocre, à tes imitations hasardeuses d'Eddy Murphy, de Nelson Monfort ou de Jean Pierre Pernaud, à tes spectacles de jongle approximatifs, à tes singularités déclinées sous toutes les sauces, et à tes piètres talents de cuisinier heureusement compensés par les bons petits plats de ta maman, ces années de vie commune resteront malgré tout un bon souvenir et je t'en remercie chaleureusement !

A titre plus personnel, je remercie profondément ma famille en Franche Comté qui malgré l'éloignement et la distance géographique me donnent l'énergie d'avancer par leurs encouragements et leur réconfort. L'épreuve la plus dure de ce travail doctoral a été leur absence, je souhaite donc leur dire à quel point je les aime.

Je souhaite aussi remercier vivement Monsieur Dimitri Komatitsch pour m'avoir donné la chance d'intégrer son équipe au sein du Laboratoire de Mécanique et d'Acoustique de Marseille avant la date de ma soutenance et pour m'y avoir accueillit si chaleureusement.

Pour conclure, je remercie tendrement celle qui partage ma vie depuis un an, ma chérie Hélène, pour tout l'amour qu'elle me porte, pour son soutien moral et pour son aide incommensurable du quotidien. Courageuse face à mon stress, à ma fatigue, à mes coups de blues, à mes galères, à l'éloignement géographique provoqué par mon départ pour Marseille, elle m'a inlassablement couvert d'attentions adorables et pour cela je la remercie du fond du cœur.

Table des matières

Introduction générale

Sommaire

Les progrès fulgurants du calcul intensif

Depuis l'ENIAC (Electronic Numerical Integrator Analyser and Computer) conçu pendant la deuxième guerre mondiale et passant pour être le premier ordinateur du monde complètement électronique jusqu'aux années 80, les supercalculateurs étaient essentiellement destinés à des applications relevant des secteurs de la défense, de la fusion nucléaire et de l'énergie. C'est en 1976 qu'est apparue Cray 1, la première machine vectorielle de Seymour Cray qui a véritablement révolutionné le calcul numérique à des fins scientifiques, permettant d'envisager un million d'opérations par seconde. Trente cinq ans plus tard, avec les progrès technologiques des circuits électroniques et des sciences de la simulation, le nombre d'opérations a été multiplié par un million et la simulation numérique s'est ouverte à un large spectre d'applications civiles. Sans se substituer complètement à l'expérience, elle a permis de réduire son usage, et donc son coût, en démultipliant ses bénéfices, devenant ainsi un outil puissant au service de la recherche. La simulation numérique est aujourd'hui devenue incontournable pour la modélisation des systèmes naturels en physique corpusculaire, chimie (simulations de dynamique moléculaire) et biologie (neurosciences computationnelles, séquençage des génomes humains, animaux, végétaux et bactériens) mais également pour la modélisation des sciences humaines et des instituts de sondage (comportement des consommateurs, stratégies de recherche d'emploi, phénomènes d'apprentissage) ou en sciences sociales pour lesquels les expériences sont difficiles, coûteuses voire impossibles. Dans tous ces domaines, l'amélioration des modèles et la nécessité d'intégrer un nombre croissant de données, ou d'utiliser un modèle couplant de nombreux phénomènes, ou encore de résoudre numériquement des équations particulièrement complexes, requièrent une grande puissance de calcul et des calculateurs de plus en plus performants.

En permettant la résolution des équations les plus complexes ou l'étude des modèles les plus sophistiqués, le calcul haute performance (High Performance Computing - HPC) ouvre de nouveaux territoires pour les entreprises, irriguant tous les secteurs, de la santé à l'énergie, de l'agronomie à la finance, du transport au bâtiment. Dans le domaine industriel et appliqué, on peut citer la géophysique (de la modélisation des tremblements de terre à la prospection sismique pour les compagnies pétrolières), ou encore l'hydrogéologie (simulation d'écoulement de l'eau) ou la simulation de réservoirs pétroliers qui permet d'optimiser l'extraction de pétrole ; l'automobile ou l'aéronautique avec des simulations de mécanique des fluides pour l'aérodynamique ou la combustion des moteurs/turbines ; les matériaux avec des simulations en modélisation moléculaire et mécanique quantique ; la fusion nucléaire avec la simulation des écoulements pour la conception des sites de stockage de déchets, l'optimisation de la sécurité en matière de confinement électromagnétique dans le cas du projet ITER ; et la finance avec la simulation de calcul de risques sur produits dérivés, le trading à haute fréquence ou le pricing de l'électricité. Mais le calcul intensif recouvre aussi de nouvelles applications de plus en plus importantes dans des domaines tels que la santé

et l'environnement. Le calcul haute performance s'impose ainsi comme un outil essentiel de la recherche scientifique, technologique et industrielle. Avec les progrès fulgurants des calculateurs, sa pratique s'est généralisée à toutes les disciplines, fondamentales et appliquées, au point de devenir en quelque sorte le troisième pilier de la science, au côté de la théorie et de l'expérimentation. Dans de nombreux domaines où l'expérimentation est impossible, comme le changement climatique ou l'astronomie, la simulation est souvent la seule solution. Dans d'autres cas, comme la recherche sur les médicaments, la science des matériaux ou la chimie, elle est envisageable, mais avec un ensemble de combinaisons hors de portée.

Puisque la complexité des phénomènes considérés va croissante, le besoin en calcul intensif ne cessera d'augmenter. Son importance n'a d'ailleurs cessé de croître avec l'augmentation des capacités de stockage et de calcul des ordinateurs et notamment avec l'avènement des calculateurs parallèles comportant de quelques dizaines à plusieurs centaines de miliers de processeurs de nos jours. Jusqu'à la fin des années 80, le calcul numérique intensif était l'apanage des machines vectorielles, spécialement conçues et optimisées pour exécuter la même instruction sur chacune des données contenues dans un tableau. Le concept sur lequel repose alors la puissance des ordinateurs est le principe du pipe line qui applique aux calculs la logique du travail à la chaîne sur des vecteurs de données. Dès le début des années 1980, les machines massivement parallèles commencent à leur faire de l'ombre et le parallélisme devient incontournable en particulier en raison d'un meilleur rapport performance/prix des microprocesseurs. Il convient donc de faire collaborer un grand nombre de processeurs entre eux. C'est Intel qui, parmi les premiers, développe efficacement ce genre d'architecture. Dans les années 90, la parallélisme devient la règle. Ces infrastructures permettent d'une part de diminuer significativement les temps de calcul dans des applications industrielles ou scientifiques complexes, et sont d'autre part très souvent scalables, c'est à dire d'une puissance extensible, dans une certaine plage, de manière à peu près proportionnelle au nombre de leurs processeurs.

Toutefois, la performance du calcul ne se résume pas à la seule technologie du supercalculateur : la modélisation, les mathématiques et l'informatique en sont des composantes majeures. La grande diversité entre les différentes classes d'architecture rend difficile l'utilisation efficace des machines parallèles, en dépit de la stabilisation vers un nombre réduit de types de machines : les machines multi-processeurs MIMD (Multiple Instructions Multiple Data) à mémoire distribuée et les machines multi-processeurs MIMD à mémoire partagée. Pour ces deux types d'architecture, chaque processeur exécute son code de manière asynchrone et indépendante mais les techniques de synchronisation dépendent de l'organisation de la mémoire. Pour une architecture à mémoire partagée, les processeurs accèdent à une mémoire commune et la synchronisation concerne tous les processeurs. En revanche, pour une architecture à mémoire distribuée, chaque processeur dispose de sa propre mémoire et n'a pas accès à celle d'autres processeurs. Un réseau d'interconnection est alors nécessaire pour échanger les informations entre processeurs sous la forme de messages, de manière synchrone ou asynchrone. Avec la progression constante des performances théoriques des supercalculateurs le calcul intensif permet de traiter de nouveaux types d'applications ; néanmoins, la complexité, la taille (en nombre de processeurs) et la hiérarchisation des architectures de calcul parallèles rendent de plus en plus cruciales des recherches sur l'optimisation des performances soulèvant des questions liées au passage à l'échelle pour les applications. Il s'agit notamment d'adapter les codes de calcul aux architectures multi-cœurs, ou aux processeurs couplés à des accélérateurs de calcul. En outre, l'évolution des architectures s'accompagne d'un accroissement de la complexité de programmation. En plus d'adapter à ces nouvelles architectures des outils de pré-traitement ou de post-traitement accompagnant le code proprement dit, il s'agit de concevoir ou de faire évoluer les bibliothèques destinées aux méthodes numériques, et éventuellement spécialisées par domaine d'application, en fonction des nouvelles architectures. Les interfaces de programmation les plus répandues, MPI pour le calcul parallèle sur des nœuds à mémoires distribuées, et OpenMP pour les nœuds à mémoires partagées, montrent leurs limites dans le cas d'architectures massivement parallèles ou hétérogènes. De nouveaux outils sont donc nécessaires, à l'image de nouveaux langages comme UPC (Unified Parallel C) ou l'architecture CUDA du constructeur américain NVIDIA pour le calcul parallèle sur processeurs graphiques. Le véritable défi revient donc à concevoir des algorithmes et des logiciels adaptés aux machines massivement parallèles

pour chaque application.

L'électromagnétisme numérique et ses applications

L'électromagnétisme numérique est une discipline relativement nouvelle qui a pu naître grâce au franchissement de la barre du calcul mégaflopique (10^6 opérations par seconde) avec les ordinateurs des années 1980, et qui bénéficie désormais de la puissance des supercalculateurs actuels de l'ordre du pétaflops, ce qui correspond à un million de milliards (10^{15}) d'opérations par seconde. Aujourd'hui, l'essor de l'électromagnétisme numérique dépend essentiellement des moyens de calcul intensif associés comme pour de nombreux autres domaines scientifiques et technologiques tels que la chimie quantique, la géophysique, la modélisation du climat ou des océans, les nanotechnologies, la biologie moléculaire et génomique, les images de synthèse et la réalité virtuelle, la physique des hautes énergies et la fusion nucléaire.

L'électromagnétisme numérique met en scène différents niveaux de modélisation. Il y a tout d'abord le modèle physique de lois de conservation de base, constitué par les équations de Maxwell. La fermeture des équations à résoudre demande ensuite une analyse phénoménologique ; les lois de comportement incorporent une globalisation de la réalité à petite échelle qui n'est pas détaillée dans le modèle final comme par exemple la structure des matériaux diélectriques absorbants. Puis vient l'étape essentielle de la simulation numérique. Pour décrire correctement le comportement de composants dont la taille ne cesse de décroître, il est de plus en plus nécessaire de faire appel à des modèles microscopiques. Ces modèles ne présentant, en général, pas de solution analytique, la méthode numérique doit être employée. Celle-ci permet, soit d'orienter le choix des approximations dans les approches analytiques, soit d'obtenir des résultats directement comparables à ceux de l'expérience. Il s'agit ici d'un apport de la simulation à la compréhension des effets physiques mis en jeu. L'optimisation permet ensuite de pouvoir intervenir sur certains paramètres pour améliorer ou optimiser le fonctionnement, le rendement, ou la réponse d'un système en maximisant (ou minimisant) des fonctions associées. L'analyse numérique et l'optimisation sont donc deux outils essentiels et complémentaires de la modélisation mathématique. Enfin les calculateurs massivement parallèles demandent de choisir un modèle de programmation pour stocker les données en mémoire et surtout gérer les communications entre processeurs de calcul.

Le champ d'application de l'électromagnétisme numérique s'est élargi à des contextes aussi variés que l'électronique, la minimisation des signatures des échos radars, les accélérateurs de particules, la magnétohydrodynamique, l'optimisation de formes d'antennes, la conception de dispositifs hyperfréquences ou la compatibilité électromagnétique qui étudie la tenue des équipements électriques et électroniques lors d'une agression par un rayonnement extérieur au système (foudre, onde intense). La santé au sens large est un autre contexte d'application de l'électromagnétisme numérique. Il s'agit par exemple de quantifier numériquement l'absorption d'un champ électromagnétique dans des tissus biologiques et les effets thermiques induits, soit pour évaluer les éventuels effets nocifs de l'exposition à ces champs [Bernardi 2000]-[Bernardi 2001]-[Clatz 2000], soit pour des besoins de planification du traitement par hyperthermie micro-onde de tumeurs cancéreuses qui nécessite de concentrer un rayonnement sur la zone malade afin de la nécroser par échauffement [Lin 2000]-[Siauve 2003].

Les équations de propagation électromagnétique, à savoir les équations de Maxwell, peuvent être formulées de différentes manières suivant les hypothèses auxquelles elles sont soumises, et ne sont par conséquent pas strictement équivalentes en terme d'approximation. On distingue en général les formulations intégrale et différentielle des équations de Maxwell exprimées dans le domaine temporel ou spectral ; à chacune de ces représentations correspond des techniques particulières de résolution. On retiendra malgré tout que plus qu'un problème de terminologie auquel tout cela peut se ramener, l'idée directrice est de déterminer les valeurs du champ électromagnétique en temps et espace, en discrétisant les opérateurs et le milieu inhérents à chaque formulation.

Les difficultés de la modélisation numérique

En dépit des avancées significatives de ces trente cinq dernières années, un grand nombre d'applications de l'électromagnétisme numérique reste hors de portée de la plupart des méthodes numériques pour la modélisation des équations de Maxwell [Reitich 2004]. Plusieurs difficultés émergent lorsque l'on tente de modéliser et de développer des méthodes de calcul pour simuler des phénomènes de propagation d'ondes électromagnétiques :

- la plupart des phénomènes électromagnétiques modélisés requièrent un domaine de calcul non borné,
- les solutions recherchées sont généralement oscillantes et la précision de toute approximation numérique dépend du pas de discrétisation qui est alors une fraction de la longueur d'onde, variable suivant les méthodes utilisées (de l'ordre du cinquième au vingtième de la longueur d'onde),
- les matériaux constituant les objets étudiés possèdent des caractéristiques électromagnétiques de plus en plus complexes qui nécessitent des algorithmes robustes pour la résolution des équations de Maxwell ; ces dernières peuvent même ne plus être linéaires pour des lois constitutives de matériaux complexes,
- la taille des problèmes qui se veulent représentatifs de la complexité des phénomènes réels est très grande. Les longueurs caractéristiques classiquement rencontrées dans les applications industrielles sont de l'ordre de quelques dizaines de longueurs d'onde mais peuvent dépasser la centaine, conduisant à des maillages qui peuvent contenir jusqu'à dix millions de mailles pour des maillages volumiques. La résolution numérique de problèmes de cette taille ne peut se faire qu'en exploitant pleinement les possibilités des calculateurs parallèles. L'algorithme généré par la méthode doit donc avoir un haut degré de parallélisme,
- les géométries compliquées induisent des singularités dans les courants surfaciques qui nécessitent souvent un traitement numérique particulier (par exemple pour les problèmes de diffraction d'onde sur une pointe). Les singularités dues aux courants filaires ne sont également pas évidentes à prendre en compte.

Résolution numérique des équations de Maxwell

Les méthodes numériques qui ont été développées ces 45 dernières années grâce aux progrès des performances informatiques, se proposent de palier les limitations des approches analytiques, et de fournir des cas de validation visant à analyser la capacité des méthodes approchées à représenter un phénomène physique réel. Le processus de validation est une étape nécessaire à la réalisation d'un outil numérique fiable. Il implique d'examiner la convergence des méthodes itératives, la consistance et la stabilité de la solution, la convergence spatiale et la convergence temporelle, de comparer les solutions aux résultats expérimentaux et d'examiner les incertitudes des modèles. Ces différents aspects de l'activité de validation forment un cadre permettant d'aboutir à des simulations de qualité y compris pour celles de type prédictives pour lesquelles les données expérimentales manquent. Pour la résolution des équations de Maxwell, il semble qu'aucune méthode numérique ne soit prédominante, le choix étant déterminé essentiellement par le type d'application considéré. Il ne s'agit pas ici de proposer une revue exhaustive de ces méthodes mais plutôt de préciser quelques caractéristiques importantes qui motivent le choix d'une méthode particulière.

On peut distinguer deux grands types de formulation des équations de Maxwell :

- la formulation en temps écrite directement à partir des équations de Maxwell-Faraday et Maxwell-Ampère.
- la formulation harmonique (ou dans le domaine fréquentiel) obtenue à l'aide d'une transformée de Fourier de la précédente.

Chacune de ces formulations a permis de construire des méthodes particulièrement bien adaptées à des problèmes spécifiques et différents de telle sorte qu'il est difficile d'affirmer une supériorité de l'une sur l'autre. Il ne faut en fait pas les considérer comme des formulations concurrentes comme l'on a trop souvent tendance à le faire mais plutôt comme complémentaires. La modélisation mathématique de ces deux types de problème est très différente et le passage d'un modèle à l'autre n'est pas toujours facile à justifier, notamment dans le cas où les coefficients sont discontinus. Les équations de Maxwell peuvent être résolues dans le domaine temporel ou bien dans le domaine fréquentiel (*i.e.* en supposant une variation harmonique en temps). De plus, la méthode numérique peut être appliquée soit sur une formulation d'équations aux dérivées partielles (EDP) des équations de Maxwell, soit sur une formulation intégrale. Le tableau 0.0.1 fournit des exemples de méthodes numériques suivant cette classification.

TABLE 0.0.1 – Classification des méthodes numériques pour les équations de Maxwell

	Domaine temporel	**Domaine fréquentiel**
Formulation EDP	DFDT	DFDF
	EFDT	EFDF
	VFDT	VFDF
	GDDT	GDDF
Formulation intégrale	MOT	MoM

Les abréviations dans le tableau 0.0.1 ont les significations suivantes :
- DF** : Différence Finie (en Domaine Temporel ou Domaine Fréquentiel)
- EF** : Elément Fini (en Domaine Temporel ou Domaine Fréquentiel)
- VF** : Volume fini (en Domaine Temporel ou Domaine Fréquentiel)
- GD** : Galerkin Discontinu (en Domaine Temporel ou Domaine Fréquentiel)
- MOT : *Marching-On in Time*
- MoM : *Method of Moments*

Le tableau 0.0.1 recense les méthodes les plus couramment utilisées dans chaque catégorie. Il existe bien sûr de nombreuses autres méthodes. La formulation temporelle des équations de Maxwell est en fait la plus générale. Elle permet en effet de modéliser à l'aide des lois constitutives des matériaux les phénomènes électromagnétiques les plus complexes qu'on peut rencontrer dans les applications réelles. Les méthodes temporelles permettent de résoudre à un moindre coût les structures complexes telles que celles composées de plusieurs couches de matériaux différents et à propriétés complexes. On peut également envisager de traiter des phénomènes électromagnétiques non-linéaires par certaines méthodes numériques. Un autre avantage indéniable est également la possibilité de réaliser en un seul calcul numérique l'étude de problèmes de propagation sur une large bande de fréquences. Elles permettent aussi de suivre l'évolution d'un signal impulsionnel en temps. En outre, les coûts informatiques (en termes de temps de calcul et d'occupation mémoire) sont en général plus faibles pour les méthodes issues d'une formulation temporelle. Les méthodes en domaine fréquentiel présentent quant à elles l'avantage d'être les mieux adaptées aux applications dont le spectre fréquentiel ne présente que quelques fréquences.

Méthodes en domaine fréquentiel : formulations intégrales

Les méthodes en domaine fréquentiel reposant sur des formulations intégrales, comme la MoM [Wang 1991], réduisent les équations volumiques à des équations surfaciques. Après discrétisation, la méthode MoM conduit à un système linéaire dense. La résolution de ce système par une méthode directe (méthode de Gauss) nécessite $O(N^3)$ opérations arithmétiques si la taille de la matrice est $N \times N$. En supposant que le nombre d'éléments par longueur d'onde est constant, N croît proportionnellement à f^2, où f est la fréquence. La complexité de résolution du système MoM par une méthode directe est alors en $O(f^6)$. Un des moyens de réduire cette complexité est de résoudre le système MoM par une méthode itérative. Le noyau numérique de base d'une méthode itérative est le produit d'une matrice

par un vecteur, lequel a une complexité en $O(N^2)$, si bien que la complexité arithmétique pour résoudre le système MoM itérativement est en $O(f^4)$ si la méthode itérative converge bien. Une réduction de cette complexité peut encore être réalisée par l'utilisation d'une méthode multipôle pour accélérer le calcul du produit matrice-vecteur moyennant quelques approximations. Dans ce cas, le système MoM peut être résolu en $O(N\ \log(N))$ opérations arithmétiques si une méthode multipôle multiniveaux est utilisée [Coifman 1993]-[Song 1995]. Une voie alternative pour réduire la complexité de la méthode MoM est d'utiliser la méthode d'optique physique (OP). Ici, les inconnues à la surface de l'objet sont calculées directement à partir du champ incident. L'interaction entre les différentes parties de la surface est donc négligée. Il s'agit d'une approximation haute fréquence. Les méthodes OP et MoM fournissent des résultats identiques lorsque la fréquence tend vers l'infini. Pour une étude récente de la méthode MoM d'un point de vue algorithmique et mise en œuvre, on pourra consulter [Bondeson 2005].

Méthodes en domaine fréquentiel : formulations EDP
Une formulation EDP des équations de Maxwell dans le domaine fréquentiel est l'équation d'Helmholtz vectorielle. Pour le champ électrique, celle-ci s'écrit :

$$\Delta \times \left(\frac{1}{\mu} \Delta \times \mathbf{E} \right) = \omega\, \varepsilon \mathbf{E} \tag{0.0.1}$$

où ω est la fréquence angulaire et ε et μ dépendent de la variable d'espace. Cette équation est en général résolue au moyen d'une méthode élément fini en raison de sa flexibilité géométrique et la possibilité d'utiliser des maillages non-structurés. Cependant, le développement de ces méthodes pour résoudre les équations de Maxwell a été relativement lent, en particulier dans le cas des méthodes d'ordre élevé. Une raison essentielle est l'obtention de solutions non physiques (apparition de modes parasites) lorsqu'un schéma élément fini classique de type Galerkin continu nodal est utilisé pour discrétiser l'équation (0.0.1). La source de ces problèmes a plusieurs interprétations parmi lesquelles une mauvaise représentation du noyau de l'opérateur rotationnel [Bossavit 1988] ou la génération de solutions qui violent les conditions de divergence, qui ne sont généralement pas directement imposées [Paulsen 1991]. Une discussion de ces questions se trouve dans [Sun 1995].

A. Bossavit a fait l'observation fondamentale que l'utilisation d'éléments finis spectraux de type $H(rot, \Omega)$ permet de palier le problème des modes parasites en préservant des principes de l'algèbre vectorielle [Bossavit 1988]-[Bossavit 1990]. Les méthodes d'éléments finis basées sur des éléments de type $H(rot, \Omega)$, aussi connues sous les noms d'éléments d'arêtes ou d'éléments de Nedelec [Nedelec 1980]-[Nedelec 1986], sont devenues rapidement l'approche dominante pour résoudre des problèmes à géométrie complexe en domaine fréquentiel. Bien que très élégantes, de telles formulations ne sont pas entièrement dénuées de points faibles : les systèmes algébriques associés sont plus grands que ceux caractérisant les éléments finis nodaux et la conformité intrinsèque de l'approximation ne facilite pas la mise en point de stratégies de résolution adaptatives (méthodes *hp*).

Méthodes en domaine temporel : formulations intégrales
Les méthodes en domaine temporel reposant sur des formulations intégrales pour les équations de Maxwell sont relativement peu utilisées. Cependant, au cours de ces dernières années, ces méthodes ont bénéficié d'un regain d'intérêt. La plupart de ces méthodes sont appelées méthodes *Marching-On In Time* (MOT). La complexité arithmétique des méthodes MOT est en $O(N_t N_s^2)$, où N_t est le nombre de pas de temps et N_s est le nombre de fonctions de base. Cette complexité peut être réduite en utilisant une technique *Plane Wave Time Domain* (PWTD) [Ergin 1998]. Les techniques PWTD s'inspirent des méthodes multipôles. La complexité d'une technique PWTD à deux niveaux est en $O(N_t N_s^{4/3} \log(N_s))$, et la complexité d'une technique PWTD multiniveaux est en $O(N_t N_s \log(N_s))$.

Un inconvénient des méthodes MOT est qu'elles sont sujettes à des instabilités [Rynne 1990]. La question a été étudiée dans [Dodson 1997] où il est montré qu'une méthode MOT pour résoudre les équations intégrales en champ magnétique peut être stabilisée en utilisant un schéma en temps implicite. Récemment, une nouvelle méthode a été développée pour éliminer l'instabilité des méthodes MOT pour des

simulations en temps longs. Cette méthode s'appelle *Marching-On In Degree* (MOD) et elle utilise des polynômes de Laguerre [Jung 2003]. En général, le coût de stockage mémoire de la méthode MOD est beaucoup plus grand que celui de la méthode MOT si le nombre de pas de temps est très grand. Une comparaison entre les méthodes MOD et MOT notamment formulation, stabilité, coût et précision, se trouve dans [Jung 2007].

Méthodes en domaine temporel : formulations EDP
Parmi les formulations EDP des équations de Maxwell en domaine temporel, on peut distinguer trois grandes classes de méthodes numériques lorsqu'il s'agit de traiter de problèmes de propagation d'ondes dans des milieux fortement hétérogènes : les méthodes de différences finies (DF), les méthodes d'éléments finis (EF), et enfin les méthodes de volumes finis (VF). La première méthode numérique efficace développée pour les équations de Maxwell a été une méthode DF proposée en 1966 par K.S. Yee [Yee 1966]. Cette méthode DFDT conserve l'énergie, propriété que l'on cherche pour les méthodes numériques appliquées aux problèmes de propagation d'ondes en domaine temporel. De plus, sa grande simplicité d'implémentation la rend peu coûteuse en taille mémoire et en temps de calcul et on peut, par exemple pour le problème de la diffraction, effectuer un calcul pour un signal large bande, et étendre par transformation de Fourier le résultat à toutes les fréquences du spectre. Malheureusement, l'utilisation d'un maillage cartésien contraint le pas d'espace à être petit pour tenir compte de la géométrie des objets représentés dès lors que ceux-ci ont des formes complexes, ce qui réduit par la même occasion le pas de temps pour les schémas explicites comme le schéma de Yee (condition de stabilité de type CFL). D'autre part, la représentation en maillages cartésiens non-uniformes engendre des ondes parasites qui perturbent les solutions numériques obtenues.

L'autre méthode importante développée une quinzaine d'années plus tard fut la méthode des EF d'arêtes [Nedelec 1980] qui fut déclinée en plusieurs versions [Monk 2003]-[Nedelec 1986]. Cette méthode présente beaucoup d'avantages, avec entre autres la faculté de traiter des maillages non-structurés et donc des géométries complexes, et de conserver l'énergie. Elle peut même être utilisée en tant que méthode d'ordre élevé [Demkowicz 2005] - [Hiptmair 2001] - [Monk 2003]. Un point faible majeur réside dans l'inversion de la matrice de masse obtenue à chaque pas de temps en domaine temporel, ce qui amène à envisager des techniques de condensation de masse [Cohen 1998]. Cet inconvénient a également conduit à l'étude des schémas de type VF pour les problèmes de propagation d'ondes en domaine temporel en raison de leur caractère hyperbolique, en s'inspirant des méthodes développées pour les problèmes issus de la mécanique des fluides. Tout comme les EF, les méthodes VF ont l'avantage de pouvoir s'adapter facilement à des maillages non-structurés. Leur particularité réside dans la formulation du flux numérique, qui peut être décentré, ce qui pour les équations de Maxwell rend le schéma diffusif [Piperno 2000] et peu dispersif, ou centré mais dans ce cas le schéma est non-diffusif mais davantage dispersif [Remaki 2000].

Le point de départ de notre étude consiste en une méthode d'éléments finis discontinus, connue sous le nom de méthode Galerkin discontinue, initialement introduite dans [Piperno 2003] - [Fezoui 2005]. Cette méthode est assez similaire aux méthodes d'éléments finis classiques, la principale différence étant la relaxation de la continuité globale de l'approximation. Le plus souvent, sa mise en œuvre repose sur une approximation polynomiale par morceaux des composantes du champ électromagnétique. Les méthodes Galerkin discontinues présentent plusieurs avantages parmi lesquels :

- elles se prêtent bien à l'utilisation de maillages non-structurés (non-orthogonaux et non uniformes) pour la discrétisation de formes géométriques complexes ;
- elles sont naturellement adaptées à la discrétisation de fonctions discontinues et à la discrétisation de milieux de propagation hétérogènes (là encore, le raffinement local joue un rôle important) ;
- elles permettent d'obtenir une précision d'ordre arbitrairement élevé (si possible, elles peuvent présenter une convergence exponentielle suivant l'ordre d'approximation comme dans les méthodes spectrales) ;
- elles autorisent une non-conformité de la discrétisation (i.e. elles permettent la prise en compte de maillages avec nœuds flottants) et de l'approximation (i.e. elles permettent un ordre d'approxima-

tion variable défini au niveau de chaque élément du maillage) pour la prise en compte de détails ou singularités géométriques (en d'autres termes, elles définissent un cadre idéal pour la mise au point de méthodes auto-adaptatives) ;

– elles sont naturellement parallélisables.

Le caractère discontinu de l'approximation impose d'avoir recours à une formulation faible locale dont le support d'intégration est l'élément (un triangle en 2D et un tétraèdre en 3D). Une intégration par parties fait alors apparaître un terme de bord dont le calcul nécessite l'introduction d'une fonction de flux numérique (similairement à ce qui est réalisé dans les méthodes volumes finis [Cioni 1993] - [Piperno 2002]). Une particularité de la méthode Galerkin discontinue introduite dans [Piperno 2003] - [Fezoui 2005] est d'utiliser une fonction de flux numérique centrée. Lorsque celle-ci est combinée à un schéma d'intégration en temps centré de type saute mouton du second ordre, on montre que l'on obtient une méthode Galerkin discontinue non-dissipative et stable sous une condition de type CFL. Cependant, lorsque les simulations numériques mettent en jeu des maillages non-uniformes (localement raffinés) constitués de petits éléments, cette condition de stabilité impose un pas de temps très pénalisant, c'est-à-dire conduisant à des temps de calcul importants voire prohibitifs pour certains problèmes tridimensionnels.

Objectifs et contributions de la thèse

En pratique, un grand nombre de simulations numériques réalistes mettent en jeu des géométries aux formes irrégulières ou des dispositifs de petite taille plongés dans des domaines de plus grande dimension. Pour ces problèmes, l'utilisation de maillages non-structurés raffinés est presque toujours souhaitable voire incontournable. La méthode Galerkin discontinue initiale en maillages simplexes introduite dans [Piperno 2003] - [Fezoui 2005] reposant sur des maillages tétrahédriques non structurés conformes et des méthodes d'interpolation $\mathbb{P}_0$ (méthode volume fini) et $\mathbb{P}_1$ (méthode Galerkin discontinue conforme basée sur une interpolation linéaire) a été récemment étendue par H. Fahs dans sa thèse [Fahs 2008] dans le but d'inclure la non-conformité de maillage et de l'ordre d'approximation. Cette thèse a permis la mise en œuvre d'une méthode Galerkin discontinue en maillages triangulaires non-structurés, d'ordre arbitrairement élevé en espace, et permettant la non-conformité de maillage et de l'ordre d'approximation, pour la résolution numérique des équations de Maxwell instationnaires bidimensionnelles, la mise en œuvre tridimensionnelle ayant fait l'objet d'une étude préliminaire.

Outre les questions de discrétisation, les stratégies de résolution auto-adaptatives font elles aussi appel à un processus de raffinement. Là encore, la possibilité de ne limiter celui-ci qu'aux régions identifiées par un critère d'erreur approprié est grandement souhaitable. Notons aussi que dans l'idéal, ces stratégies auto-adaptatives cherchent à combiner un raffinement du maillage (h-raffinement) dans les zones de moindre régularité de la solution avec un enrichissement de l'ordre d'approximation (p-enrichissement) là où la solution est régulière. On parle alors de méthodes hp-adaptatives.

L'objectif général de notre étude est de mettre au point différentes stratégies permettant d'améliorer l'efficacité de la méthode Galerkin discontinue introduite dans [Fezoui 2005] vers le développement d'une méthode hp-adaptative en maillages simplexes non-structurés pour la résolution numérique des équations de Maxwell instationnaires. D'un point de vue numérique, on vise ici deux objectifs complémentaires :

– améliorer la flexibilité de l'approximation polynomiale pour faciliter l'élaboration d'une méthode hp-adaptative.

– augmenter la précision de l'approximation temporelle par une stratégie de pas de temps local pour pemettre la non-conformité temporelle entre les domaines.

Sur un plan plus informatique, on s'intéresse à l'adaptation des noyaux de calcul d'une méthode Galerkin discontinue d'ordre élevé aux caractéristiques des architectures de calcul haute performance modernes combinant cœur de calcul (CPU) et accélérateur matériel (GPU).

Le plan de la thèse est le suivant :

Cadre mathématique et numérique

Le chapitre 1 est consacré à une présentation du système de Maxwell qui modélise les phénomènes électromagnétiques. Il installe le cadre mathématique et numérique en détaillant la méthode Galerkin discontinue en domaine temporel (méthode GDDT) pour la résolution numérique de ce système d'équations.

Etude numérique comparative d'interpolations polynomiales

Dans la méthode GDDT introduite au chapitre 1, l'approximation des composantes du champ électromagnétique au sein d'un élément du maillage repose sur une interpolation polynomiale nodale de type Lagrange. L'intégration en temps des équations semi-discrétisées est réalisé au moyen d'un schéma saute-mouton précis au second ordre. Dans la construction d'une méthode GDDT d'ordre arbitrairement élevé, le choix d'une technique d'interpolation polynomiale doit s'appuyer sur plusieurs critères parmi lesquels le caractère nodal ou modal de l'interpolation, la nécessité de formules de quadrature pour le calcul d'intégrales élémentaires, le caractére local ou non des calculs mettant en jeu un simplexe de dimension inférieure (typiquement, dans les calcul d'intégrales sur la frontière du simplexe primal) et le caractère hiérarchique ou non de l'interpolation en vue de faciliter la mise en œuvre d'une méthode où l'ordre d'approximation est variable en espace. Eventuellement, on peut aussi souhaiter que l'approximation préserve des propriétés liées au modèle d'équations différentielles résolu comme la positivité de certaines quantités physiques. Un premier objectif de ce chapitre est d'évaluer en détail différentes méthodes d'interpolation polynomiale en association avec la formulation GDDT introduite au chapitre 1. Par ailleurs, l'obtention d'une méthode Galerkin discontinue d'ordre arbitrairement élevé nécessite l'utilisation d'un schéma d'intégration en temps de précision compatible avec l'ordre d'approximation en espace. Dans ce chapitre, on se propose aussi d'évaluer l'apport de schémas temporels de type saute-mouton et Runge-Kutta du quatrième ordre en lien avec les différentes techniques d'interpolation polynomiales considérées.

Méthodes d'ordre élevé avec pas de temps local

L'utilisation d'un schéma d'intégration en temps explicite conditionnellement stable sur un maillage raffiné en espace suppose, par le biais de la condition CFL, que le pas de temps global est fixé par la taille du plus petit élément. Ainsi, la modélisation numérique de problèmes de propagation d'ondes électromagnétiques faisant intervenir des géométries complexes ou nécessitant la discrétisation de milieux de propagation hétérogènes, de matériaux réalistes ou de structures de petite taille sur des maillages non-uniformes entraine inévitablement une dégradation des perfomances et donc une perte d'efficacité du schéma numérique. Nous nous intéressons dans le chapitre 3 à une technique de pas de temps local qui permet d'adapter le pas de temps à la taille des mailles et proposons une nouvelle méthode Galerkin discontinu d'ordre arbitrairement élevé en espace dans l'esprit de celles récemment proposées par [Diaz 2009]-[Grote 2009] pour la résolution numérique des équations de Maxwell du premier ordre. Des expériences numériques en 1D et en 2D sont réalisées afin d'illustrer l'utilité de la méthode à pas de temps local en terme de performance.

Calcul haute performance sur architecture multi-GPU

On présente dans le chapitre 4 une méthodologie numérique adaptée au calcul haute performance. Le système des équations de Maxwell est discrétisé par une méthode Galerkin discontinue formulée en maillages tétraédriques et reposant sur une approximation polynomiale d'ordre élevée des composantes du champ électromagnétique au sein d'un élément. Les équations semi-discrétisées sont intégrées en temps au moyen d'un schéma saute mouton du second ordre. La méthodologie numérique ainsi développée est adaptée aux architectures de calcul haute performance modernes comprenant des cartes accélératrices de type GPU, en adoptant une stratégie hybride combinant un modèle de programmation SPMD à grain grossier pour la parallélisation inter-GPU et une modèle SIMD à grain fin pour la parallélisation intra-GPU. On rappelle comment en quelques années les GPU se sont transformés de puces dédiés aux fonctions graphiques en processeurs massivement parallèles (Single Instruction Multiple threads) que l'on programme en langage haut-niveau (C/C++). A travers l'exemple de l'architecture CUDA (Compute Unified Device Architecture) de NVidia, on présente le modèle d'exécution et de programmation de ces processeurs et

de leurs limitations. Enfin, les bénéfices en termes de performance résultant de l'accélération multi-GPU sont démontrés par la réalisation de simulations sur plusieurs systèmes multi-GPU.

Liste des publications issues de cette thèse

Articles acceptés

- **Performance evaluation of a multi-GPU enabled finite element method for computational electromagnetics**
  T. Cabel, J. Charles and S. Lanteri
  Lecture Notes in Computer Science (LNCS) - Springer, 2012

Rapports de recherche INRIA

- **Multi-GPU acceleration of a DGTD method for modelling human exposure to electromagnetic waves**
  T. Cabel, J. Charles and S. Lanteri
  INRIA research report, RR-7592, April 2011
- **Etude numérique d'interpolations polynomiales dans une méthode Galerkin discontinue pour la résolution numérique des équations de Maxwell instationnaires 1D**
  J. Charles, L. Fezoui and S. Lanteri
  INRIA research report, RR-7177, August 2010

Cette thèse a été partiellement financée par un contrat de recherche entre l'équipe projet NACHOS et le centre CEA DAM, CESTA de Bordeaux (Direction des applications militaires du Commissariat à l'énergie atomique et aux énergies alternatives français). L'objectif général de cette étude est le développement d'un solveur couplé Vlasov-Maxwell combinant la méthode DGTD-$\mathbb{P}_p$ d'ordre élevé pour la résolution des équations de Maxwell en domaine temporel sur des maillages tétraédriques développée dans l'équipe projet NACHOS et une méthode Particle-In-Cell. Le solveur DGTD-PIC résultant servira à étudier l'intéraction des ondes électromagnétiques avec des faisceaux de particules chargées et sera utilisé pour des études de vulnérabilité électrique de la chambre expérimentale du système Laser Mégajoule.

Chapitre 1

Cadre mathématique et numérique

1.1 Les équations de Maxwell

Nous rappelons ici les quatre équations fondamentales, appelées équations de Maxwell ou encore équations de Maxwell-Lorentz, qui décrivent complètement l'évolution spatio-temporelle des champs électrique et magnétique et leur interdépendance dans le cadre de la physique classique. Ces équations traduisent, sous forme locale, différents théorèmes (Gauss, Ampère, Faraday) qui décrivaient l'électromagnétisme avant que Maxwell ne les réunisse sous forme d'équations intégrales. Nous présenterons dans un premier temps la théorie microscopique fondamentale qui conduit aux équations de Maxwell dans le vide en présence de sources, qui peuvent être des charges ponctuelles et/ou leurs courants électriques microscopiques associés si ces charges sont en mouvement dans le référentiel d'étude ; puis on abordera la théorie macroscopique nécessitant l'introduction des champs $\mathbf{D}$ et $\mathbf{H}$.

1.1.1 Equations de Maxwell 3D

Equation de Maxwell-Gauss
Cette équation locale exprime la divergence du champ électrique en fonction de la densité de la charge électrique :

$$\mathrm{div}(\mathbf{E}) = \frac{\rho}{\varepsilon_0}. \tag{1.1.1}$$

L'équation locale de Maxwell-Gauss est identique en régime variable ou en statique. Le théorème de Gauss qui en découle a donc un caractère universel et permet de lier le flux du champ électrique à travers une surface fermée à la charge intérieure à cette surface :

$$\oint_S \mathbf{E}.d\mathbf{S} = \frac{Q_{int}}{\varepsilon_0},$$

où S est une surface fermée arbitraire, appelée surface de Gauss, Q_{int} est la charge électrique totale intérieure à cette surface et ε_0 est la permittivité électrique du vide.

Equation de Maxwell-Thomson
Cette équation locale relie la divergence du champ magnétique au terme de source, ici identiquement nul :

$$\mathrm{div}(\mathbf{B}) = 0. \tag{1.1.2}$$

Elle traduit le fait expérimental suivant : il n'existe pas de monopôle magnétique, c'est à dire une source ponctuelle de champ magnétique, analogue de la charge électrique ponctuelle pour le champ électrique. Le champ d'induction magnétique peut toutefois être décrit en terme de courants et l'équivalent d'un dipôle magnétique serait une boucle infiniment petite. L'absence de charge magnétique entraîne la nullité du flux d'induction magnétique à travers toute surface fermée S :

$$\oint_S \mathbf{B}.d\mathbf{S} = 0.$$

L'équation de Maxwell-Thomson est valable pour tous les régimes et pour tous les milieux et indique simplement que le champ magnétique **B** est à flux conservatif.

Equation de Maxwell-Faraday
Cette équation locale qui est applicable pour tous les régimes et pour tous les milieux, traduit le phénomène fondamental d'induction électromagnétique découvert par Faraday :

$$\mathbf{rot}(\mathbf{E}) = -\frac{\partial \mathbf{B}}{\partial t}. \tag{1.1.3}$$

On remarque que les variations temporelles du champ magnétique **B** sont nécessairement liées à la présence d'un champ électrique **E**. Sa forme intégrale est la loi de Faraday :

$$e = -\frac{\partial \Phi}{\partial t},$$

où e désigne la force électromotrice d'induction traversant un circuit fermé Γ et Φ le flux magnétique à travers ce même circuit Γ. Cette loi est le pilier de l'induction électromagnétique qui stipule que les variations temporelles du flux magnétique à travers un circuit induisent l'apparition d'une force électromotrice qui engendre dans un circuit fermé des courants appelés courants induits.

Equation de Maxwell-Ampère
On retrouve le lien entre le champ magnétique et ses sources dans l'équation de Maxwell-Ampère : la présence du champ magnétique **B** est due à l'existence de courants électriques (densité de courant **j**) et à la dépendance en temps du champ électrique **E** :

$$\mathbf{rot}(\mathbf{B}) = \mu_0 \, \mathbf{j} + \varepsilon_0 \mu_0 \frac{\partial \mathbf{E}}{\partial t}, \tag{1.1.4}$$

où μ_0 correspond à la perméabilité magnétique du vide. L'équation de Maxwell-Ampère peut être réécrite de la manière suivante :

$$\mathbf{rot}(\mathbf{B}) = \mu_0(\mathbf{j} + \mathbf{j_D}) \quad \text{avec} \quad \mathbf{j_D} = \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t},$$

où $\mathbf{j_D}$ représente la densité de courant de déplacement, courant fictif qui traduit une équivalence avec les variations temporelles du champ électrique. Le bilan macroscopique de l'équation locale de Maxwell-Ampère nous donne le théorème d'Ampère généralisé selon lequel le flux du champ magnétique **B** à travers un contour fermé Γ est égal à la somme des intensités traversant Γ :

$$\oint_{\Gamma} \mathbf{B}.\mathbf{dl} = I + I_D,$$

où I est l'intensité du courant véritable et I_D celle du courant de déplacement à travers le contour fermé Γ. Ce théorème montre que les sources du champ magnétique **B** sont les courants électriques véritables et ceux de déplacement, ces derniers résultant des variations temporelles du champ électrique.

Milieux diélectriques
A l'exception du vide, tous les milieux matériels dans lesquels peuvent se propager des ondes électromagnétiques sont dispersifs c'est à dire des milieux où la célérité des ondes qui se propagent dépend de leur fréquence. On distingue essentiellement trois types de milieux :

- les conducteurs (principalement les métaux),
- les plasmas qui sont des gazs ionisés (comportant donc des ions positifs et des électrons),
- les diélectriques (on appelle ainsi des milieux isolants tels que le verre et de nombreux plastiques).

Dans un conducteur, les charges électriques sont libres de se déplacer dans le matériau. Dans les métaux, ces charges sont des électrons des couches externes des atomes qui forment un gaz d'électrons libres. Nous savons qu'un champ électromagnétique variable ne pénètre que faiblement (sur une distance infime

appelée épaisseur de peau) dans le métal et se disperse. Dans un conducteur parfait, la conductivité est infinie, l'épaisseur de peau est alors nulle et l'onde ne peut pas s'y propager.

Dans les plasmas, la conductivité électrique est plus faible que dans les métaux ; elle est essentiellement liée au mouvement des électrons, les cations beaucoup plus lourds étant moins mobiles. Le passage d'une onde électromagnétique dans un plasma va engendrer le mouvement des électrons et des cations, créant ainsi des courants qui vont eux-mêmes modifier le champ électromagnétique incident. La propagation d'une onde électromagnétique est donc différente de celle qu'elle est dans le vide.

Lorsque le milieu considéré est un diélectrique, il n'existe pas de charges électriques libres susceptibles de se déplacer de façon macroscopique, mais des charges liées qui peuvent s'écarter légèrement de leur position d'équilibre. Les milieux diélectriques sont ceux pour lesquels les atomes ne peuvent pas libérer de charges qui participent à la conduction du courant électrique.

La polarisation

Dans un milieu matériel, les charges en présence appartiennent à des structures complexes : atomes, molécules d'un gaz, d'un liquide ou d'un solide. Sous l'influence d'une excitation électrique extérieure, elles réagissent *collectivement* : on dit que le milieu *se polarise*. Bien que les diélectriques, matériaux isolants, ne possèdent pas de charges de conduction, l'application d'un champ $\mathbf{E}$ extérieur aboutit à la création de charges de polarisation par la déformation des répartitions des charges à l'intérieur des diélectriques. Plus précisement, un diélectrique polarisé acquiert un moment dipolaire électrique qu'on peut décrire *macroscopiquement* par une densité volumique $\mathbf{P}$, appelée *vecteur polarisation*, telle que dans tout volume élémentaire du diélectrique apparaisse le moment élémentaire :

$$d\mathbf{p} = \mathbf{P} d\tau.$$

La matière diélectrique polarisée peut être considérée comme du vide, siège d'une densité de courant volumique fictif. On parle de *courants liés*, par opposition aux courants véritables, dits libres. Cette *densité de courants liés*, présente en tout point de la matière diélectrique polarisée, s'écrit :

$$\mathbf{j_p} = \frac{\partial \mathbf{P}}{\partial t},$$

avec $\mathbf{P}$ le vecteur polarisation.

De manière similaire, dans un milieu diélectrique de vecteur polarisation $\mathbf{P}$, il existe des courants de polarisation microscopiques caractérisés au niveau mésoscopique par la *densité volumique de charges liées* :

$$\rho_\mathbf{p} = -\mathrm{div}(\mathbf{P}).$$

On va modéliser le milieu diélectrique par du vide dans lequel sont introduites les densités de courant $\mathbf{j_p} = \dfrac{\partial \mathbf{P}}{\partial t}$ et de charges $\rho_\mathbf{p} = -\mathrm{div}(\mathbf{P})$. Les équations de Maxwell (1.1.1-1.1.4) s'écrivent alors :

$$\begin{cases} \mathbf{rot}(\mathbf{B}) &= \mu_0(\mathbf{j} + \mathbf{j_p}) + \varepsilon_0\mu_0 \dfrac{\partial \mathbf{E}}{\partial t}, \\ \mathbf{rot}(\mathbf{E}) &= -\dfrac{\partial \mathbf{B}}{\partial t}, \\ \mathrm{div}(\mathbf{E}) &= \dfrac{\rho + \rho_p}{\varepsilon_0}, \\ \mathrm{div}(\mathbf{B}) &= 0. \end{cases} \tag{1.1.5}$$

Seules les équations de Maxwell-Gauss et Maxwell-Ampère sont modifiées. Si l'on introduit le vecteur polarisation $\mathbf{P}$, celles-ci deviennent :

$$\left\{\begin{array}{l} \mathrm{div}(\mathbf{E}) = \dfrac{\rho - \mathrm{div}(\mathbf{P})}{\varepsilon_0} \\ \mathbf{rot}(\mathbf{B}) = \mu_0(\mathbf{j} + \dfrac{\partial \mathbf{P}}{\partial t}) + \varepsilon_0\mu_0 \dfrac{\partial \mathbf{E}}{\partial t} \end{array}\right. \Rightarrow \left\{\begin{array}{l} \mathrm{div}(\varepsilon_0\mathbf{E} + \mathbf{P}) = \rho \\ \mathbf{rot}(\mathbf{B}) = \mu_0(\mathbf{j} + \dfrac{\partial}{\partial t}(\varepsilon_0\mathbf{E} + \mathbf{P})) \end{array}\right.$$

Propriétés diélectriques
Quand on applique un champ électrique dans un matériau diélectrique, les charges subissent une force, mais sont retenues par les forces de cohésion internes de l'atome ou de la molécule, et ne peuvent donc se déplacer que légèrement par rapport à leur position d'équilibre. Il y a alors formation de dipôles induits dans le matériau, qui créent un champ de polarisation, fonction du champ électrique appliqué. Les effets électriques sont alors représentés par un *champ de déplacement* définit comme suit :

$$\mathbf{D} = \varepsilon_0\mathbf{E} + \mathbf{P}.$$

Les équations de Maxwell prennent alors la forme :

$$\left\{\begin{array}{lcl} \mathbf{rot}(\mathbf{B}) & = & \mu_0(\mathbf{j} + \dfrac{\partial \mathbf{D}}{\partial t}), \\ \mathbf{rot}(\mathbf{E}) & = & -\dfrac{\partial \mathbf{B}}{\partial t}, \\ \mathrm{div}(\mathbf{D}) & = & \rho, \\ \mathrm{div}(\mathbf{B}) & = & 0. \end{array}\right. \tag{1.1.6}$$

Dans le vide, et également de façon approchée dans l'air, il n'y a pas de polarisation $\mathbf{P}$, et on a par conséquent $\mathbf{D} = \varepsilon_0\mathbf{E}$.

Propriétés magnétiques
Les propriétés magnétiques des matériaux résultent d'une propriété quantique de l'électron, appellée *moment magnétique de spin*, qui peut être positif ou négatif. Dans la grande majorité des éléments, le nombre de spins positifs est égal à celui des spins négatifs : leurs effets se compensent exactement et le milieu considéré ne possède pas de propriété magnétique. Dans certains matériaux de transition, dits *ferromagnétiques*, le nombre d'électrons à spin positif et à spin négatif est différent. La résultante n'est plus nulle et entraîne une aimantation $\mathbf{M}$. On définit le champ magnétique $\mathbf{H}$ au moyen de la relation :

$$\mathbf{H} = \frac{1}{\mu_0}(\mathbf{B} - \mathbf{M}).$$

Dans le vide, l'air et les matériaux non ferromagnétiques, l'aimantation $\mathbf{M}$ est nulle ou négligeable et on a alors $\mathbf{B} = \mu_0\mathbf{H}$.

Forme locale ou différentielle
En tout point de l'espace qui n'est pas situé sur une surface de séparation entre deux milieux, les équations de Maxwell s'écrivent :

$$\left\{\begin{array}{lcl} \mathbf{rot}(\mathbf{H}) & = & \dfrac{\partial \mathbf{D}}{\partial t} + \mathbf{j}, \\ \mathbf{rot}(\mathbf{E}) & = & -\dfrac{\partial \mathbf{B}}{\partial t}, \\ \mathrm{div}(\mathbf{D}) & = & \rho, \\ \mathrm{div}(\mathbf{B}) & = & 0. \end{array}\right. \tag{1.1.7}$$

Sous cette forme, dite locale ou différentielle, on remarque un comportement symétrique des quatre champs. Seule l'absence de sources magnétiques rompt cette symétrie et l'introduction d'une densité

volumique de courant magnétique $\mathbf{m}$ ainsi que d'une densité de charge magnétique libre ϱ permettrait de symétriser totalement le système de Maxwell.

1.1.2 Discontinuités du champ électromagnétique et conditions aux limites

Dans les équations de Maxwell n'apparaissent que les densités de charges ou de courants réparties dans un volume. Lorsque le modèle de sources du champ impose que ces charges ou courants ne soient présents que sur des courbes ou des surfaces, les équations de Maxwell doivent être remplacées par des relations dites *de passage* qui expriment la discontinuité du champ électromagnétique à la traversée de telles singularités. L'écriture des équations de Maxwell au sens des distributions permet alors d'expliciter de manière générale les conditions aux limites ou d'interface associées à ce type de problème.

Equations au sens des distributions
Soit une distribution $\mathbf{T_A}$ de $\mathbb{R}^3$ associée au champ de vecteur $\mathbf{A}$ dérivable au sens des fonctions vectorielles dans le complémentaire d'une surface de discontinuité S orientée et de normale $\mathbf{n}$. On rappelle que la divergence et le rotationnel d'une distribution vectorielle $\mathbf{T_A}$ s'écrivent :

$$\begin{cases} \mathrm{div}(\mathbf{T_A}) &= \mathrm{div}(\mathbf{A}) + \mathbf{n}.[\mathbf{A}]_S\,\delta_S, \\ \mathbf{rot}(\mathbf{T_A}) &= \mathbf{rot}(\mathbf{A}) + \mathbf{n}\times[\mathbf{A}]_S\,\delta_S, \end{cases}$$

où δ_S est la distribution de Dirac sur la surface S et $[\mathbf{A}]_S$ le saut de $\mathbf{A}$ à travers la surface S. En utilisant ces relations ainsi que les distributions associées aux grandeurs intervenant dans les équations de Maxwell, on aboutit à :

$$\begin{cases} \dfrac{\partial \mathbf{D}}{\partial t} - \mathbf{rot}(\mathbf{H}) - \mathbf{n}\times[\mathbf{H}]_S\,\delta_S &= -\mathbf{j} - \mathbf{j_S}\delta_S, \\ \dfrac{\partial \mathbf{B}}{\partial t} + \mathbf{rot}(\mathbf{E}) + \mathbf{n}\times[\mathbf{E}]_S\,\delta_S &= 0, \\ \mathrm{div}(\mathbf{D}) + \mathbf{n}.[\mathbf{D}]_S\,\delta_S &= \rho + \rho_S\,\delta_S, \\ \mathrm{div}(\mathbf{B}) + \mathbf{n}.[\mathbf{B}]_S\,\delta_S &= 0, \end{cases} \tag{1.1.8}$$

où la densité de charge ρ, prise elle aussi au sens des distributions, peut se décomposer comme la somme d'une densité volumique et d'une densité superficielle répartie sur la surface S : $T_\rho = \rho + \rho_S\,\delta_S$. On peut alors déduire de (1.1.8), par identification, les relations de sauts normaux et tangentiels grâce aux équations de Maxwell prises au sens des fonctions :

$$\begin{cases} \mathbf{n}\times[\mathbf{H}]_S &= \mathbf{j_S}, \\ \mathbf{n}\times[\mathbf{E}]_S &= 0, \\ \mathbf{n}.[\mathbf{D}]_S &= \rho + \rho_S\delta_S, \\ \mathbf{n}.[\mathbf{B}]_S &= 0. \end{cases} \tag{1.1.9}$$

Conditions d'interface
Nous venons de décrire les conditions d'interface à travers une surface de discontinuité qui peut éventuellement porter des sources de courants. En l'absence de courant, les composantes tangentielles des champs électrique et magnétique sont continues à travers toute surface d'après les relations :

$$\begin{cases} \mathbf{n}\times[\mathbf{H}]_S &= 0, \\ \mathbf{n}\times[\mathbf{E}]_S &= 0. \end{cases} \tag{1.1.10}$$

On remarquera que les relations (1.1.10) sont toujours vérifiées y compris en présence de matériaux différents d'indices quelconques. En revanche, les deux autres relations montrent que, même en l'absence de charge, les composantes normales des champs électrique et magnétique ne sont, en général, pas continues :

$$\begin{cases} \mathbf{n}.[\varepsilon \mathbf{E}]_S &= 0, \\ \mathbf{n}.[\mu \mathbf{H}]_S &= 0. \end{cases} \tag{1.1.11}$$

Ce type de relations s'applique également pour la définition des conditions aux limites, notamment pour les bords métalliques.

Conditions aux limites pour une frontière métallique parfaitement conductrice

Nous assimilons tout au long de cette étude les parois métalliques au modèle idéal et fictif du conducteur parfait. Nous considérons donc que le champ électromagnétique est nul à l'intérieur du métal. Soient $\mathbf{E_S}$ et $\mathbf{H_S}$ les champs électrique et magnétique sur la surface extérieure du métal, les relations de saut (1.1.9) deviennent :

$$\begin{cases} \mathbf{n} \times \mathbf{H_s} &= \mathbf{j_S}, \\ \mathbf{n} \times \mathbf{E_S} &= 0, \\ \mathbf{n}.\mathbf{E_S} &= \dfrac{\rho_S}{\varepsilon}, \\ \mathbf{n}.\mathbf{H_S} &= 0. \end{cases} \tag{1.1.12}$$

On en déduit en particulier qu'à la surface d'un conducteur parfait, le champ électrique est normal et le champ magnétique tangent.

Conditions aux limites pour une frontière artificielle

Numériquement, nous devons nous restreindre à un domaine de calcul borné alors que les ondes électromagnétiques se propagent dans un domaine infini. De nombreux auteurs ont cherché à définir des problèmes aux limites associés, bien posés. Des conditions aux limites totalement absorbantes (ou transparentes, i.e. ne générant aucune onde parasite à la frontière artificielle) ont été établies mais peuvent difficilement être implémentées numériquement car elles ne sont pas locales à la fois en temps et en espace. Cependant, une série de conditions aux limites absorbantes, qui sont une approximation des conditions transparentes, sont maintenant bien connues pour l'équation des ondes et le système de Maxwell. Nous utilisons ici la condition absorbante d'ordre un de Silver-Müller pour une frontière placée dans le vide :

$$\mathbf{n} \times \mathbf{E} = -z_0 \, \mathbf{n} \times (\mathbf{n} \times \mathbf{H}), \tag{1.1.13}$$

où z_0 représente l'impédance caractéristique du vide définie par : $z_0 = \sqrt{\dfrac{\mu_0}{\varepsilon_0}}$.

1.1.3 Formulation conservative

Une formulation conservative classique des équations de Maxwell en fonction des quatre champs de vecteur **E**, **H**, **D** et **B** est donnée par (1.1.7). Nous allons utiliser un schéma conservatif, en particulier pour le traitement des discontinuités des champs. Afin de résoudre numériquement ce système, nous devons prendre en compte les lois constitutives des matériaux dont la forme la plus générale est donnée par :

$$\mathbf{D} = \varepsilon_0 \mathbf{E} + \mathbf{P} \qquad \mathbf{B} = \mu_0 (\mathbf{H} + \mathbf{M}) \qquad \mathbf{J} = \sigma(\mathbf{E}).$$

La polarisation $\mathbf{P}$ et l'aimantation $\mathbf{M}$ dépendent des champs électrique et magnétique. Dans le cas le plus général, on a des fonctions $\mathbf{P}(\mathbf{E},\mathbf{H})$ et $\mathbf{M}(\mathbf{E},\mathbf{H})$ et on parle de milieux chiraux, bi-isotropes et bi-anisotropes. Dans les milieux couramment rencontrés en électrotechnique, les relations prennent habituellement la forme $\mathbf{P}(\mathbf{E})$ et $\mathbf{M}(\mathbf{H})$. Les dépendances peuvent être complexes et non linéaires. Heureusement, beaucoup de matériaux couramment utilisés ont des propriétés pratiquement linéaires et des modèles linéaires simples suffisent pour décrire leurs propriétés.

Dans le cas d'un milieu linéaire, isotrope, et éventuellement hétérogène, la polarisation $\mathbf{P}$ est donc une fonction linéaire de $\mathbf{E}$ et, de même, l'aimantation $\mathbf{M}$ est directement proportionnelle à $\mathbf{H}$. On peut dans ce cas écrire que :

$$\begin{cases} \mathbf{D} = \varepsilon(\mathbf{x})\mathbf{E}, \\ \mathbf{B} = \mu(\mathbf{x})\mathbf{H}. \end{cases} \tag{1.1.14}$$

Les fonctions ε et μ de la variable d'espace $\mathbf{x}$ sont scalaires. Dans l'ensemble de cette étude, nous ne considérerons en fait que des interfaces entre des matériaux isotropes et homogènes (i.e. ε et μ constants par morceaux). Nous omettons souvent par la suite la dépendance en fonction de la variable d'espace de la permittivité et de la perméabilité afin d'alléger l'écriture.

Notons que les lois de Gauss électrique et magnétique $\mathrm{div}(\mathbf{D}) = \rho$ et $\mathrm{div}(\mathbf{B}) = 0$ sont redondantes dans le modèle continu (1.1.7) pour une donnée initiale vérifiant (1.1.14) et en considérant de plus l'équation de conservation de la charge :

$$\frac{\rho}{t} + \mathrm{div}(\mathbf{j}) = 0.$$

On peut ainsi ne considérer que les deux premières équations de (1.1.7) qui s'écrivent alors pour la formulation conservative en fonction du champ électromagnétique ${}^t(\mathbf{E},\mathbf{H})$:

$$\begin{cases} \varepsilon \dfrac{\partial \mathbf{E}}{\partial t} - \mathbf{rot}\mathbf{H} &= -\mathbf{j}, \\ \mu \dfrac{\partial \mathbf{H}}{\partial t} + \mathbf{rot}\mathbf{E} &= 0. \end{cases} \tag{1.1.15}$$

Considérons un ouvert Ω, borné et régulier de $\mathbb{R}^3$, de frontière $\Gamma = \partial\Omega = \Gamma^a \cup \Gamma^m$ (avec $\Gamma^a \cap \Gamma^m = \varnothing$), où Γ^m désigne une frontière métallique et Γ^a le bord artificiel du domaine de calcul, comme représenté sur la figure 1.1.

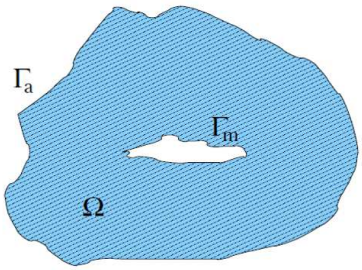


Figure 1.1 – Domaine de calcul

On peut alors formuler le système (1.1.15) sous la forme suivante :

$$\left\{ \begin{array}{rcl} \varepsilon \dfrac{\partial \mathbf{E}}{\partial t} + N_x \dfrac{\partial \mathbf{H}}{\partial x} + N_y \dfrac{\partial \mathbf{H}}{\partial y} + N_z \dfrac{\partial \mathbf{H}}{\partial z} & = & -\sigma \mathbf{E}, \\ \mu \dfrac{\partial \mathbf{H}}{\partial t} - N_x \dfrac{\partial \mathbf{E}}{\partial x} - N_y \dfrac{\partial \mathbf{E}}{\partial y} - N_z \dfrac{\partial \mathbf{E}}{\partial z} & = & 0, \end{array} \right.$$

avec :

$$\mathbf{E} = \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix} \quad \text{et} \quad \mathbf{H} = \begin{pmatrix} H_x \\ H_y \\ H_z \end{pmatrix},$$

où l' on a supposé que le milieu de propagation est conducteur avec $\mathbf{j} = \sigma \mathbf{E}$, et pour lequel les matrices (antisymétriques) N_x, N_y et N_z sont données par :

$$N_x = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix} \quad , \quad N_y = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad , \quad N_z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Si l'on note :

$$G_l = \begin{pmatrix} 0_{3\times 3} & N_l \\ -N_l & 0_{3\times 3} \end{pmatrix} \quad \text{avec} \quad l \in x, y, z,$$

alors les équations de Maxwell (1.1.15) s'écrivent sous forme vectorielle de la façon suivante :

$$\partial_t (Q\mathbf{W}) + G_x \partial_x \mathbf{W} + G_y \partial_y \mathbf{W} + G_z \partial_z \mathbf{W} + K\mathbf{W} = 0, \tag{1.1.16}$$

où :

$$Q = \begin{pmatrix} \varepsilon & 0 \\ 0 & \mu \end{pmatrix} \ , \ \mathbf{W} = \begin{pmatrix} \mathbf{E} \\ \mathbf{H} \end{pmatrix} \ , \ K = \begin{pmatrix} \sigma & 0 \\ 0 & 0 \end{pmatrix},$$

soit sous forme condensée (puisque ε et μ sont supposés indépendants du temps) :

$$Q\mathbf{W}_t + \nabla \cdot F(\mathbf{W}) + K\mathbf{W} = 0, \tag{1.1.17}$$

et :

$$F(\mathbf{W}) = \begin{pmatrix} F_x(\mathbf{W}) \\ F_y(\mathbf{W}) \\ F_z(\mathbf{W}) \end{pmatrix},$$

avec :

$$F_x(\mathbf{W}) = \begin{bmatrix} 0_{3\times3} & N_x \\ -N_x & 0_{3\times3} \end{bmatrix} \mathbf{W} = \begin{pmatrix} 0 \\ H_z \\ -H_y \\ 0 \\ -E_z \\ E_y \end{pmatrix},$$

$$F_y(\mathbf{W}) = \begin{bmatrix} 0_{3\times3} & N_y \\ -N_y & 0_{3\times3} \end{bmatrix} \mathbf{W} = \begin{pmatrix} -H_z \\ 0 \\ H_x \\ E_z \\ 0 \\ -E_x \end{pmatrix},$$

$$F_z(\mathbf{W}) = \begin{bmatrix} 0_{3\times3} & N_z \\ -N_z & 0_{3\times3} \end{bmatrix} \mathbf{W} = \begin{pmatrix} H_y \\ -H_x \\ 0 \\ -E_y \\ E_x \\ 0 \end{pmatrix}.$$

1.1.4 Hyperbolicité du système de Maxwell

Le caractère hyperbolique est intrinsèque au système de Maxwell et a une interprétation physique. Les ondes et l'énergie associée se propagent en temps fini suivant des directions particulières. La principale application de cette propriété est la construction de schémas décentrés, précis à la fois en temps et en espace, qui tiennent naturellement compte de la direction de propagation des ondes. Toutefois, l'utilisation de flux décentrés introduit malheureusement de la diffusion numérique, et par conséquent la solution est altérée après un nombre de pas de temps significatif. Ainsi, nous ferons le choix, dans cette étude, d'un schéma reposant sur une formulation à flux centrés, permettant une conservation de l'énergie.

Rappelons que le champ électromagnétique $\mathbf{W}$ vérifie l'équation suivante :

$$Q\partial_t(\mathbf{W}) + G_x\partial_x\mathbf{W} + G_y\partial_y\mathbf{W} + G_z\partial_z\mathbf{W} + K\mathbf{W} = 0.$$

Si on introduit $G_{\mathbf{n}} = n_xG_x + n_yG_y + n_zG_z$ où $\mathbf{n}$ désigne un vecteur non-nul de $\mathbb{R}^3$, le fait que le système des équations de Maxwell (1.1.15) soit hyperbolique signifie que la matrice de flux $\bar{G}_{\mathbf{n}} = G_{\mathbf{n}}Q^{-1}$ est diagonalisable dans $\mathbb{R}^6$ et l'on a :

$$\begin{cases} \bar{G}_{\mathbf{n}} &= T_{\mathbf{n}}\Lambda_{\mathbf{n}}{T_{\mathbf{n}}}^{-1}, \\ \bar{G}^{\pm}_{\mathbf{n}} &= T_{\mathbf{n}}\Lambda^{\pm}_{\mathbf{n}}{T_{\mathbf{n}}}^{-1}, \\ \Lambda^{+}_{\mathbf{n}} &= \mathrm{diag}(\max(\Lambda_{\mathbf{n},k}, 0)), \\ \Lambda^{-}_{\mathbf{n}} &= \mathrm{diag}(\min(\Lambda_{\mathbf{n},k}, 0)), \end{cases} \tag{1.1.18}$$

où $\Lambda_{\mathbf{n},k}$, $k = 1, \ldots, 6$ désignent les valeurs propres de la matrice de flux $\bar{G}_{\mathbf{n}}$:

$$\Lambda_{\mathbf{n}} = \mathrm{diag}(-c, -c, 0, c, c) \quad , \quad c = \frac{1}{\sqrt{\varepsilon\mu}}.$$

1.1.5 Redimensionnement

Il est classique d'exprimer les permittivité et perméabilité des matériaux par des quantités adimensionnelles : la permittivité relative ou *c*onstante diélectrique et la perméabilité relative, normalisées par rapport à un milieu de référence :

$$\varepsilon(\mathbf{x}) = \varepsilon_0\, \varepsilon_r(\mathbf{x}) \quad \text{et} \quad \mu(\mathbf{x}) = \mu_0\, \mu_r(\mathbf{x}).$$

Le vide est choisi comme milieu de référence, car il est linéaire, homogène, isotropique, et avec réponse instantanée, et car avec ces propriétés, la permittivité du vide devient une constante :

$$\varepsilon_0 = 8,854187 \cdot 10^{-12} \quad \mathrm{F} \cdot \mathrm{m}^{-1}$$

La vitesse relative de la lumière est maintenant notée :

$$c_r = \frac{1}{\sqrt{\varepsilon_r \mu_r}}.$$

En particulier dans le vide, on a $\varepsilon_r = \mu_r = c_r = 1$.

On procède au redimensionnement des équations de Maxwell en effectuant les changements de variables suivants :

$$\begin{cases} \tilde{t} = c_0\, t, \\ \\ \tilde{H} = z_0\, H, \end{cases} \tag{1.1.19}$$

où $c_0 = 1/\sqrt{\varepsilon_0 \mu_0}$ et $z_0 = \sqrt{\mu_0/\varepsilon_0} = \mu_0 c_0$ est l'impédance du vide.

Après redimensionnement, on obtient le système des équations de Maxwell suivant. Notons que pour simplifier les notations, le $\tilde{}$ a été omis sur les quantités redimensionnées :

$$\begin{cases} \varepsilon \dfrac{\partial \mathbf{E}}{\partial t} - \mathbf{rot}\mathbf{H} &= -z_0 \sigma \mathbf{E}, \\ \\ \mu \dfrac{\partial \mathbf{H}}{\partial t} + \mathbf{rot}\mathbf{E} &= 0, \end{cases} \tag{1.1.20}$$

où ε et μ définissent des quantités relatives.

1.1.6 Conditions aux limites et initiales

Sur la frontière Γ, on impose deux types de conditions aux limites, $\mathbf{n}$ étant la normale sortante à Γ :

$$\begin{cases} \text{- sur } \Gamma^m & : \mathbf{n} \times \mathbf{E} = 0, \\ \text{- sur } \Gamma^a & : \mathbf{n} \times \mathbf{E} + z\mathbf{n} \times (\mathbf{n} \times \mathbf{H}) = \mathbf{n} \times \mathbf{E}^\infty + z\mathbf{n} \times (\mathbf{n} \times \mathbf{H}^\infty), \end{cases} \tag{1.1.21}$$

où $z = \sqrt{\dfrac{\mu}{\varepsilon}}$ et ${}^t(\mathbf{E}^\infty, \mathbf{H}^\infty)$ est un champ incident donné. La première relation définit une condition de réflexion totale (le champ $\mathbf{E}$ ne pénètre pas la frontière Γ^m comme dans le cas d'un métal parfait). La seconde relation est la condition absorbante de Silver-Müller qui est une approximation du premier ordre de la condition de rayonnement en domaine infini.

Enfin, on complète la formulation du problème continu par la donnée de conditions initiales :

$$\mathbf{E}_0(\mathbf{x}) = \mathbf{E}(\mathbf{x}, 0) \;\text{ et }\; \mathbf{H}_0(\mathbf{x}) = \mathbf{H}(\mathbf{x}, 0)\ ,\ \forall \mathbf{x} \in \Omega.$$

1.2 Discrétisation par une méthode GD à flux centré

1.2.1 Les méthodes GDDT

La méthode Galerkin discontinue (GD) a été introduite pour la première fois en 1973 par Reed et Hill [Reed 1973] pour résoudre le problème de transport neutronique. Cette méthode repose sur une base de fonctions discontinues d'un élément du maillage à l'autre. L'ordre d'interpolation peut varier arbitrairement dans chaque élément. La méthode peut être vue comme une approche éléments finis pour laquelle aucune continuité n'est imposée entre éléments, ou une approche volumes finis d'ordre élevé. Par ailleurs, la discontinuité de l'approximation permet de n'imposer aucune contrainte sur le maillage et les discrétisations non-conformes sont donc autorisées. De plus, les matrices de masse obtenues sont locales à une élément ce qui permet de s'affranchir de la question cruciale de l'inversion d'une matrice de masse globale typique des méthodes d'éléments finis classiques. Ces propriétés font des méthodes Galerkin discontinues des candidates idéales pour mettre au point des stratégies de résolution *hp*-adaptatives. Enfin la méthode GD est fréquemment utilisée pour résoudre les systèmes différentiels hyperboliques non-linéaires de la mécanique des fluides compressibles [Remacle 2003] - [Remacle 2001] - [Xin 2006] mais son application à la résolution des équations de Maxwell instationnaires est en revanche plus récente [Kabakian 2004] - [Fezoui 2005] - [Min 2005] - [Ji 2005]. Pour ces équations, des méthodes GDDT d'ordre élevé ont été développées en maillages tétraédriques [Hesthaven 2002]-[Fezoui 2005] et en maillages hexaédriques [Cohen 2006].

Les méthodes GDDT peuvent être basées sur des flux décentrés [Hesthaven 2002] - [Hesthaven 2004a] - [Kopriva 2000] - [Warburton 2000]. Par exemple, Cockburn et Shu [Cockburn 1989] utilisent une formulation GD en espace, combinée à un schéma en temps de type Runge-Kutta pour discrétiser des systèmes d'EDP hyperboliques. Kopriva *et al.* [Kopriva 2000] ont développé une méthode GD qui combine des éléments spectraux avec un schéma de Runge-Kutta peu coûteux, d'ordre quatre, en utilisant des maillages conformes et non-conformes. Dans leurs travaux, Hesthaven et Warburton [Warburton 2000] - [Hesthaven 2002] utilisent une méthode *Runge-Kutta Discontinuous Galerkin* (RKDG) basée sur une interpolation polynomiale nodale pour l'approximation locale du champ électromagnétique. Comme tout schéma basé sur des flux décentrés, celui-ci ne conserve aucune énergie électromagnétique discrète. Chen *et al.* [Chen 2005] ont développé une méthode RKDG pour les équations de Maxwell en formulation du premier ordre, qui permet une convergence d'ordre élevé en espace et en temps, en utilisant un schéma de Runge-Kutta particulier (*Strong Stability Preserving Runge-Kutta* (SSP-RK)). En utilisant des polynômes locaux à divergence nulle, Cockburn *et al.* [Cockburn 2004] ont développé une méthode GDDT pour les équations de Maxwell du premier ordre, qui conserve localement la divergence discrète. Pour les équations de Maxwell en domaine fréquentiel, les méthode GD sur des maillages simplexes ont été étudiées par Hesthaven et Warburton [Hesthaven 2004b], Dolean *et al.* [Dolean 2008], et Houston *et al.* [Houston 2005] - [Buffa 2007], tandis que des maillages non-conformes orthogonaux sont considérés dans [Houston 2003] - [Houston 2004].

Piperno *et al.* ont développé une méthode GDDT qui combine des flux centrés avec un schéma saute-mouton [Fezoui 2005]. Ce schéma conserve une énergie électromagnétique discrète et préserve les relations de divergence (dans un certain sens) en l'absence de source, mais la vitesse de convergence de ce schéma semble sous-optimale. Canouet *et al.* [Canouet 2005] ont proposé une nouvelle méthode de type Galerkin discontinue basée sur un espace d'approximation local $\mathbb{P}^1_{\text{div}}$, un schéma saute-mouton pour l'intégration en temps et un schéma centré pondéré pour le calcul des flux. Malheureusement ce schéma peut dans certains cas conduire à des solutions incorrectes dans le cas d'un maillage localement raffiné de façon non-conforme. Pour pallier ce problème, les auteurs proposent un schéma hybride $\mathbb{P}^1_{\text{div}}/\mathbb{P}^2_{\text{div}}$ avec des flux centrés. Les résultats numériques montrent clairement l'intérêt de ce type de méthode en maillages non-conformes pour la conception d'antennes. Néanmoins, bien que vérifiée numériquement, la stabilité de la méthode n'est pas étudiée théoriquement.

Dans cette section, nous présentons, dans un cadre général, la méthode GDDT non-dissipative proposée

dans [Fezoui 2005] pour la résolution des équations de Maxwell en maillages non-structurés conformes. Cette méthode combine l'utilisation d'une approximation centrée pour l'évaluation des flux aux interfaces entre éléments voisins du maillage, à un schéma d'intégration en temps de type saute-mouton d'ordre deux. Cette méthode servira de point de départ pour notre étude.

1.2.2 Formulation faible

Le principe général des méthodes de type Galerkin discontinu consiste à rechercher les inconnues numériques du problème dans des espaces de dimension finie (non nécessairement polynomiaux), sous la forme d'une combinaison linéaire de fonctions de base locales de ces sous-espaces dans chaque cellule. La solution approchée par une méthode GD peut être discontinue à l'interface entre deux cellules voisines, ce qui amène à définir une fonction de flux numérique pour calculer les intégrales aux interfaces des cellules de contrôle. La principale différence entre une méthode GD et une méthode Galerkin classique (méthode élément fini continue - EF) réside dans la relaxation de la continuité globale de l'approximation. Dans le premier cas, les systèmes sont locaux *i.e.* leur taille ne dépend que de la dimension du problème physique et du degré d'interpolation choisi alors que dans le second cas il faut assembler les matrices locales pour obtenir la matrice globale dont le stockage et surtout l'inversion requiert de bonnes propriétés concernant le profil et le conditionnement.

Comme pour toute méthode de type Galerkin, l'approximation est tout d'abord définie localement (au niveau des éléments de la triangulation) sur la base d'une représentation polynomiale. Ensuite, parce que l'on ne cherche pas à imposer la continuité globale de l'approximation, on définit une formulation faible au niveau d'un élément τ_i.

On multiplie (1.1.17) par une fonction test scalaire φ (on suppose $\mathbf{j} = 0$ pour simplifier la présentation, soit $K \equiv 0$), on intégre sur τ_i et on intègre par parties :

$$\begin{aligned} (1.1.17) \Rightarrow \quad & \int_{\tau_i} Q\mathbf{W}_t \, \varphi \, d\mathbf{x} + \int_{\tau_i} (\nabla \cdot F(\mathbf{W})) \, \varphi \, d\mathbf{x} = 0, \\ \Leftrightarrow \quad & \int_{\tau_i} Q\mathbf{W}_t \, \varphi \, d\mathbf{x} - \int_{\tau_i} \nabla\varphi \cdot F(\mathbf{W}) d\mathbf{x} + \int_{\partial\tau_i} (F(\mathbf{W}) \cdot \mathbf{n}) \, \varphi \, d\sigma = 0. \end{aligned} \tag{1.2.1}$$

Soit $\mathcal{P}_i = \mathbb{P}_{p_i}[\tau_i]$ l'espace des fonctions polynomiales de degré au plus p_i sur τ_i. Les degrés de liberté locaux sont notés $\mathbf{W}_{ij} \in \mathbb{R}^6$. Soit $\phi_i = (\varphi_{i1}, \varphi_{i2}, \cdots, \varphi_{id_i})$ une base locale de $\mathcal{P}_i$ et $\mathbf{W}_i$ la projection L_2-orthogonale de $\mathbf{W}$ sur $\mathcal{P}_i$:

$$\mathbf{W}_i(\mathbf{x}) = \sum_{j=1}^{d_i} \mathbf{W}_{ij} \varphi_{ij}(\mathbf{x}). \tag{1.2.2}$$

Soient $\mathcal{V}_i = \{j | \tau_i \cap \tau_j \neq 0\}$ l'ensemble des éléments voisins de τ_i ; $a_{ij} = \tau_i \cap \tau_j$ la face commune à τ_i et τ_j et $\mathbf{n}_{ij}$ le vecteur normal sortant à a_{ij} dirigé de τ_i vers τ_j. En injectant l'expression (1.2.2) dans les intégrales volumiques de (1.2.1) et en tenant compte de la linéarité de $F(\mathbf{W})$ nous obtenons :

$$\begin{aligned} (1.2.1) \Rightarrow \quad & \int_{\tau_i} Q(\mathbf{W}_i)_t \, \varphi \, d\mathbf{x} - \int_{\tau_i} \nabla\varphi \cdot F(\mathbf{W}_i) d\mathbf{x} \quad + \quad \int_{\partial\tau_i} (F(\mathbf{W}) \cdot \mathbf{n}) \, \varphi \, d\sigma = 0, \\ \Leftrightarrow \quad & Q_i \int_{\tau_i} (\mathbf{W}_i)_t \, \varphi \, d\mathbf{x} - \int_{\tau_i} \nabla\varphi \cdot F(\mathbf{W}_i) d\mathbf{x} \quad + \quad \sum_{j \in \mathcal{V}_i} \int_{a_{ij}} (F(\mathbf{W}) \cdot \mathbf{n}_{ij}) \, \varphi \, d\sigma \\ & \qquad\qquad\qquad\qquad\qquad\qquad + \quad \int_{\partial\tau_i \cap \Gamma} (F(\mathbf{W}) \cdot \mathbf{n}) \, \varphi \, d\sigma = 0, \end{aligned} \tag{1.2.3}$$

où :

$$Q_i = \begin{bmatrix} \varepsilon_i I_{3\times 3} & 0_{3\times 3} \\ 0_{3\times 3} & \mu_i I_{3\times 3} \end{bmatrix}.$$

Les quantités ε_i et μ_i sont des valeurs des paramètres électromagnétiques constantes sur l'élément τ_i. Puisque la représentation locale de $\mathbf{W}$ est discontinue d'un élément à un autre, un traitement particulier doit être introduit pour l'évaluation de l'intégrale de bord sur la face a_{ij}. Dans le contexte des méthodes volumes finis, on parle de *flux numérique*. Nous utilisons ici un flux numérique basé sur un schéma centré :

$$F(\mathbf{W})|_{a_{ij}} \approx \frac{F(\mathbf{W}_i) + F(\mathbf{W}_j)}{2}. \tag{1.2.4}$$

Par suite :

$$\begin{aligned}
(1.2.3) \quad \Rightarrow \quad & Q_i \int_{\tau_i} (\mathbf{W}_i)_t \, \varphi \, d\mathbf{x} - \int_{\tau_i} \nabla\varphi \cdot F(\mathbf{W}_i) d\mathbf{x} \\
& + \frac{1}{2} \sum_{j\in\mathcal{V}_i} \int_{a_{ij}} (F(\mathbf{W}_i)\cdot \mathbf{n}_{ij}) \, \varphi \, d\sigma + \frac{1}{2} \sum_{j\in\mathcal{V}_i} \int_{a_{ij}} (F(\mathbf{W}_j)\cdot \mathbf{n}_{ij}) \, \varphi \, d\sigma \\
& + \int_{\partial\tau_i\cap\Gamma} (F(\mathbf{W})\cdot \mathbf{n}) \, \varphi \, d\sigma = 0, \\
\Leftrightarrow \quad & Q_i \int_{\tau_i} (\mathbf{W}_i)_t \, \varphi \, d\mathbf{x} - \int_{\tau_i} \left(\sum_{k\in\{x,y,z\}} \partial_k\varphi \, F_k(\mathbf{W}_i) \right) d\mathbf{x} \\
& + \frac{1}{2} \sum_{j\in\mathcal{V}_i} \int_{a_{ij}} \left(\sum_{k\in\{x,y,z\}} n_{ij}^k F_k(\mathbf{W}_i) \right) \varphi \, d\sigma \\
& + \frac{1}{2} \sum_{j\in\mathcal{V}_i} \int_{a_{ij}} \left(\sum_{k\in\{x,y,z\}} n_{ij}^k F_k(\mathbf{W}_j) \right) \varphi \, d\sigma + \int_{\partial\tau_i\cap\Gamma} (F(\mathbf{W})\cdot \mathbf{n}) \, \varphi \, d\sigma = 0,
\end{aligned} \tag{1.2.5}$$

avec $\mathbf{n}_{ij} = {}^t(n_{ij}^x, n_{ij}^y, n_{ij}^z)$. En posant :

$$G_k = \begin{bmatrix} 0_{3\times 3} & N_k \\ -N_k & 0_{3\times 3} \end{bmatrix},$$

on obtient finalement :

$$\begin{aligned}
(1.2.5) \quad \Rightarrow \quad & Q_i \int_{\tau_i} (\mathbf{W}_i)_t \, \varphi \, d\mathbf{x} - \int_{\tau_i} \left(\sum_{k\in\{x,y,z\}} \partial_k\varphi \, G_k \mathbf{W}_i \right) d\mathbf{x} \\
& + \frac{1}{2} \sum_{j\in\mathcal{V}_i} \int_{a_{ij}} G_{ij} \mathbf{W}_i \varphi \, d\sigma \\
& + \frac{1}{2} \sum_{j\in\mathcal{V}_i} \int_{a_{ij}} G_{ij} \mathbf{W}_j \varphi \, d\sigma + \int_{\partial\tau_i\cap\Gamma} (F(\mathbf{W})\cdot \mathbf{n}) \, \varphi \, d\sigma = 0,
\end{aligned} \tag{1.2.6}$$

où :

$$\left\{\begin{array}{lcl} G_{\mathbf{n}} & = & \displaystyle\sum_{k\in\{x,y,z\}} n^k G_k = \left[\begin{array}{cc} 0_{3\times3} & N_{\mathbf{n}} \\ -N_{\mathbf{n}} & 0_{3\times3} \end{array}\right], \\ G_{ij} & = & G_{\mathbf{n}_{ij}}, \\ N_{\mathbf{n}} & = & \displaystyle\sum_{k\in\{x,y,z\}} n^k N_k, \\ N_{ij} & = & N_{\mathbf{n}_{ij}}. \end{array}\right. \tag{1.2.7}$$

On notera que N_{ij} est une matrice antisymétrique et par suite, G_{ij} est une matrice symétrique.

1.2.3 Traitement numérique des conditions aux limites

Les conditions aux limites (1.1.21) sont prises en compte dans la formulation faible (1.2.6) en évaluant le terme de bord portant sur les faces f de $\partial\tau_i$ telles que $f \in \partial\tau_i \cap \Gamma$ par un flux numérique approprié. On choisit ici d'utiliser un schéma centré (1.2.4) et pour cela, il est nécessaire de définir des états $\mathbf{W}_j$ *artificiels* appropriés. Ainsi, ces faces sont aussi (abusivement) référencées par a_{ij} où τ_j est un élément fictif.

Frontières métalliques. Notons $\mathcal{F}_m$ l'ensemble des faces frontières métalliques du maillage $\mathcal{T}_h$ et $\mathcal{F}_m^i$ l'ensemble des faces métalliques du tétraèdre τ_i. Si $a_{ij} \in \mathcal{F}_m^i$ alors nous posons :

$$\mathbf{W}_j = \left[\begin{array}{cc} -I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{array}\right] \mathbf{W}_i.$$

et $\mathbf{W}_{|a_{ij}} = \dfrac{\mathbf{W}_i + \mathbf{W}_j}{2}$ est tel que :

$$\mathbf{W}_{|a_{ij}} = \left[\begin{array}{cc} 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{array}\right] \mathbf{W}_i.$$

Frontières absorbantes. Notons $\mathcal{F}_a$ l'ensemble des faces frontières absorbantes du maillage $\mathcal{T}_h$ et $\mathcal{F}_a^i$ l'ensemble des faces absorbantes du tétraèdre τ_i. Si $a_{ij} \in \mathcal{F}_a^i$ alors nous posons :

$$\mathbf{W}_j = \left[\begin{array}{cc} 0_{3\times3} & z_i N_{ij} \\ -z_i^{-1} N_{ij} & 0_{3\times3} \end{array}\right] \mathbf{W}_i + \left[\begin{array}{cc} I_{3\times3} & -z_i N_{ij} \\ z_i^{-1} N_{ij} & I_{3\times3} \end{array}\right] \mathbf{W}_i^{\infty}.$$

où $\mathbf{W}_i^{\infty}$ est la projection de $\mathbf{W}^{\infty}$ sur τ_i avec $\mathbf{W}^{\infty}$ un champ incident donné. Alors, $\mathbf{W}_{|a_{ij}} = \dfrac{\mathbf{W}_i + \mathbf{W}_j}{2}$ est tel que :

$$\mathbf{W}_{|a_{ij}} = \frac{1}{2}\left[\begin{array}{cc} I_{3\times3} & z_i N_{ij} \\ -z_i^{-1} N_{ij} & I_{3\times3} \end{array}\right] \mathbf{W}_i + \frac{1}{2}\left[\begin{array}{cc} I_{3\times3} & -z_i N_{ij} \\ z_i^{-1} N_{ij} & I_{3\times3} \end{array}\right] \mathbf{W}_i^{\infty}.$$

A partir de maintenant, nous désignons par $\mathcal{F}^i$ l'ensemble des faces de τ_i et on a $\mathcal{F}^i = \mathcal{F}_d^i \bigcup \mathcal{F}_a^i \bigcup \mathcal{F}_m^i$ où $\mathcal{F}_d^i$, $\mathcal{F}_a^i$ et $\mathcal{F}_m^i$ représentent respectivement l'ensemble des faces internes, absorbantes et métalliques du tétraèdre τ_i. On a alors :

$$\sum_{j\in\mathcal{V}_i} \int_{a_{ij}} G_{ij}\mathbf{W}_j\,\varphi\,d\sigma = \sum_{a_{ij}\in\mathcal{F}_d^i} \int_{a_{ij}} G_{ij}\mathbf{W}_j\varphi\,d\sigma + \sum_{a_{ij}\in\mathcal{F}_a^i} \int_{a_{ij}} G_{ij}\mathbf{W}_j\varphi\,d\sigma + \sum_{a_{ij}\in\mathcal{F}_m^i} \int_{a_{ij}} G_{ij}\mathbf{W}_j\varphi\,d\sigma.$$

Le calcul des flux internes (i.e. des termes de bord pour $a_{ij} \in \mathcal{F}_d^i$) est immédiat. Ensuite :

– pour $a_{ij} \in \mathcal{F}_a^i$:

$$\begin{aligned} G_{ij}\mathbf{W}_j &= G_{ia}\begin{bmatrix} 0_{3\times3} & z_i N_{ia} \\ -z_i^{-1}N_{ia} & 0_{3\times3} \end{bmatrix}\mathbf{W}_i + G_{ia}\begin{bmatrix} I_{3\times3} & -z_i N_{ia} \\ z_i^{-1}N_{ia} & I_{3\times3} \end{bmatrix}\mathbf{W}_i^\infty, \\ &= \begin{bmatrix} -z_i^{-1}N_{ia}^2 & 0_{3\times3} \\ 0_{3\times3} & -z_i N_{ia}^2 \end{bmatrix}\mathbf{W}_i + \begin{bmatrix} z_i^{-1}N_{ia}^2 & N_{ia} \\ -N_{ia} & z_i N_{ia}^2 \end{bmatrix}\mathbf{W}_i^\infty, \\ &\equiv P_{ia}\mathbf{W}_i + P_{i\infty}\mathbf{W}_i^\infty, \end{aligned}$$

où l'on a introduit la notation $_{ia}$ pour rappeler que la face a_{ij} est ici une face $f \in \partial\tau_i \cap \Gamma^a$. Ainsi, dans l'expression de N_{ia}, $\mathbf{n}_{ia}$ désigne la normale à la face f extérieure à τ_i.

– pour $a_{ij} \in \mathcal{F}_m^i$:

$$G_{ij}\mathbf{W}_j = G_{im}\begin{bmatrix} -I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{bmatrix}\mathbf{W}_i = \begin{bmatrix} 0_{3\times3} & N_{im} \\ N_{im} & 0_{3\times3} \end{bmatrix}\mathbf{W}_i \equiv P_{im}\mathbf{W}_i,$$

où l'on a introduit la notation $_{im}$ pour rappeler que la face a_{ij} est ici une face $f \in \partial\tau_i \cap \Gamma^m$. Ainsi, dans l'expression de N_{im}, $\mathbf{n}_{im}$ désigne la normale à la face f extérieure à τ_i.

A ce stade, une remarque s'impose concernant le traitement des termes de bord associés aux faces absorbantes. Le système des équations de Maxwell écrit sous forme conservative :

$$\begin{cases} \varepsilon\dfrac{\partial \mathbf{E}}{\partial t} - \mathbf{rot}(H) &= 0, \\ \mu\dfrac{\partial \mathbf{H}}{\partial t} + \mathbf{rot}(\mathbf{E}) &= 0, \end{cases} \tag{1.2.8}$$

avec $\mathbf{D} = \varepsilon\mathbf{E}$ et $\mathbf{B} = \mu\mathbf{H}$, est hyperbolique. Par suite, la matrice :

$$\bar{G}_{\mathbf{n}} = G_{\mathbf{n}}Q^{-1} = \begin{bmatrix} 0_{3\times3} & \dfrac{N_{\mathbf{n}}}{\mu} \\ -\dfrac{N_{\mathbf{n}}}{\varepsilon} & 0_{3\times3} \end{bmatrix},$$

est diagonalisable et on montre que l'on a la décomposition suivante :

$$\begin{aligned} \bar{G}_{\mathbf{n}} &= \bar{G}_{\mathbf{n}}^+ + \bar{G}_{\mathbf{n}}^-, \\ &= \frac{c}{2}\begin{bmatrix} -N_{\mathbf{n}}^2 & z^{-1}N_{\mathbf{n}} \\ -zN_{\mathbf{n}} & -N_{\mathbf{n}}^2 \end{bmatrix} + \frac{c}{2}\begin{bmatrix} N_{\mathbf{n}}^2 & z^{-1}N_{\mathbf{n}} \\ -zN_{\mathbf{n}} & N_{\mathbf{n}}^2 \end{bmatrix}. \end{aligned}$$

avec $c = \dfrac{1}{\sqrt{\varepsilon\mu}}$. On en déduit :

$$\begin{cases} \bar{G}_{\mathbf{n}}^\pm &= G_{\mathbf{n}}^\pm Q^{-1}, \\ G_{\mathbf{n}}^+ &= \dfrac{1}{2}\begin{bmatrix} -z^{-1}N_{\mathbf{n}}^2 & N_{\mathbf{n}} \\ -N_{\mathbf{n}} & -zN_{\mathbf{n}}^2 \end{bmatrix}, \\ G_{\mathbf{n}}^- &= \dfrac{1}{2}\begin{bmatrix} z^{-1}N_{\mathbf{n}}^2 & N_{\mathbf{n}} \\ -N_{\mathbf{n}} & zN_{\mathbf{n}}^2 \end{bmatrix}. \end{cases}$$

On constate donc que $P_{i\infty}\mathbf{W}_i^\infty = 2G_{\mathbf{n}_{i\infty}}^-\mathbf{W}_i^\infty$ est un terme qui traduit l'information entrante à la frontière artificielle Γ^a issue du champ incident $\mathbf{W}_i^\infty$. Il est aussi intéressant de faire le lien entre le flux centré :

$$G_{ij}\left(\frac{\mathbf{W}_i + \mathbf{W}_j}{2}\right),$$

et la condition de Silver-Müller (seconde relation de (1.1.21)) en utilisant les matrices $G_{ij}^\pm$.

1.2.4 Formulation matricielle

Commençons par introduire les notations suivantes :

$$\left\{\begin{array}{lcl} \Phi_i & = & \displaystyle\int_{\tau_i} {}^t\phi_i\phi_i d\mathbf{x}, \\ \Phi_i^k & = & \displaystyle\int_{\tau_i} {}^t\left(\partial_k\phi_i\right)\phi_i d\mathbf{x}, \\ \Phi_{ii} & = & \displaystyle\int_{a_{ij}} {}^t\phi_i\phi_i d\sigma, \\ \Phi_{ij} & = & \displaystyle\int_{a_{ij}} {}^t\phi_i\phi_j d\sigma. \end{array}\right.$$

où :

- $\phi_i = (\varphi_{i1}, \varphi_{i2}, \cdots, \varphi_{id_i})$ (ϕ_i est un vecteur $d_i \times 1$),
- Φ_i est une matrice $d_i \times d_i$ symétrique définie positive,
- Φ_i^k est une matrice $d_i \times d_i$,
- Φ_{ii} est *a priori* une matrice $d_i \times d_i$ symétrique définie positive,
- Φ_{ij} est *a priori* une matrice rectangulaire quelconque si $d_i \neq d_j$ (Φ_{ij} est une matrice $d_i \times d_i$ symétrique positive si $d_i = d_j$).

Posons $\mathbb{E}_i = (\mathbf{E}_{i1}, \mathbf{E}_{i2}, \cdots, \mathbf{E}_{id_i})$, $\mathbb{H}_i = (\mathbf{H}_{i1}, \mathbf{H}_{i2}, \cdots, \mathbf{H}_{id_i})$ ($\mathbb{E}_i$ et $\mathbb{H}_i$ sont des matrices rectangulaires $3 \times d_i$). Si dans (1.2.6) φ est remplacée par φ_{ij} pour $1 \leq j \leq d_i$ alors nous obtenons :

$$\forall \tau_i \in \mathcal{T}_h \ : \left\{\begin{array}{lcl} Q_{\varepsilon,i}(\mathbb{E}_i)_t\,{}^t\Phi_i & - & \displaystyle\sum_{k=1}^{3} N_k \mathbb{H}_i\,{}^t\Phi_i^k \\ & + & \displaystyle\frac{1}{2}\sum_{a_{ij}\in\mathcal{F}^i} N_{ij}\mathbb{H}_i\,{}^t\Phi_{ii} + \frac{1}{2}\sum_{a_{ij}\in\mathcal{F}_d^i} N_{ij}\mathbb{H}_j\,{}^t\Phi_{ij} \\ & + & \displaystyle\sum_{a_{ij}\in\mathcal{F}_m^i} N_{im}\mathbb{H}_i\,{}^t\Phi_{ij} - \sum_{a_{ij}\in\mathcal{F}_a^i} z_i^{-1}N_{ia}^2\mathbb{E}_i\,{}^t\Phi_{ij} \\ & = & \displaystyle -\sum_{a_{ij}\in\mathcal{F}_a^i}\left(z_i^{-1}N_{ia}^2\mathbb{E}_i^\infty + N_{ia}\mathbb{H}_i^\infty\right){}^t\Phi_{ij}, \\ Q_{\mu,i}(\mathbb{H}_i)_t\,{}^t\Phi_i & + & \displaystyle\sum_{k=1}^{3} N_k \mathbb{E}_i\,{}^t\Phi_i^k \\ & - & \displaystyle\frac{1}{2}\sum_{a_{ij}\in\mathcal{F}^i} N_{ij}\mathbb{E}_i\,{}^t\Phi_{ii} - \frac{1}{2}\sum_{a_{ij}\in\mathcal{F}_d^i} N_{ij}\mathbb{E}_j\,{}^t\Phi_{ij} \\ & + & \displaystyle\sum_{a_{ij}\in\mathcal{F}_m^i} N_{im}\mathbb{E}_i\,{}^t\Phi_{ij} - \sum_{a_{ij}\in\mathcal{F}_a^i} z_i N_{ia}^2\mathbb{H}_i\,{}^t\Phi_{ij} \\ & = & \displaystyle -\sum_{a_{ij}\in\mathcal{F}_a^i}\left(-N_{ia}\mathbb{E}_i^\infty + z_i N_{ia}^2\mathbb{H}_i^\infty\right){}^t\Phi_{ij}. \end{array}\right.$$

Les inconnues étant habituellement sous forme de vecteurs, nous réécrivons ces équations en vectorisant les matrices $\mathbb{E}_i$ et $\mathbb{H}_i$. Dans ce qui suit, $\mathbf{E}_i$ et $\mathbf{H}_i$ désignent maintenant des vecteurs $3d_i \times 1$ des degrés de liberté locaux $\mathbf{E}_{ik}$ et $\mathbf{H}_{ik}$ pour $k = 1, \ldots, d_i$ associés à la cellule τ_i. On obtient alors :

$$
\forall \tau_i \in \mathcal{T}_h \ : \ \left\{ \begin{array}{rcl}
\mathcal{X}_{\varepsilon,i} \dfrac{d\mathbf{E}_i}{dt} & - & \displaystyle\sum_{k=1}^{3} \mathcal{X}_i^k \mathbf{H}_i + \frac{1}{2} \sum_{a_{ij} \in \mathcal{F}^i} \mathcal{X}_{ij} \mathbf{H}_i + \frac{1}{2} \sum_{a_{ij} \in \mathcal{F}_d^i} \mathcal{X}_{ij} \mathbf{H}_j \\
& + & \displaystyle\sum_{a_{ij} \in \mathcal{F}_m^i} \mathcal{X}_{im} \mathbf{H}_i - \sum_{a_{ij} \in \mathcal{F}_a^i} \mathcal{X}_{ia}^E \mathbf{E}_i = - \sum_{a_{ij} \in \mathcal{F}_a^i} \mathcal{X}_{i\infty}^E \mathbf{W}_i^\infty, \\
\mathcal{X}_{\mu,i} \dfrac{d\mathbf{H}_i}{dt} & + & \displaystyle\sum_{k=1}^{3} \mathcal{X}_i^k \mathbf{E}_i - \frac{1}{2} \sum_{a_{ij} \in \mathcal{F}^i} \mathcal{X}_{ij} \mathbf{E}_i - \frac{1}{2} \sum_{a_{ij} \in \mathcal{F}_d^i} \mathcal{X}_{ij} \mathbf{E}_j \\
& + & \displaystyle\sum_{a_{ij} \in \mathcal{F}_m^i} \mathcal{X}_{im} \mathbf{E}_i - \sum_{a_{ij} \in \mathcal{F}_a^i} \mathcal{X}_{ia}^H \mathbf{H}_i = - \sum_{a_{ij} \in \mathcal{F}_a^i} \mathcal{X}_{i\infty}^H \mathbf{W}_i^\infty,
\end{array} \right. \tag{1.2.9}
$$

avec :

$$
\left\{ \begin{array}{rcl}
\mathcal{X}_{\varepsilon,i} & = & \begin{bmatrix} (\Phi_i)_{11} Q_{\varepsilon,i} & (\Phi_i)_{12} Q_{\varepsilon,i} & \cdots & (\Phi_i)_{1d_i} Q_{\varepsilon,i} \\ (\Phi_i)_{21} Q_{\varepsilon,i} & (\Phi_i)_{22} Q_{\varepsilon,i} & \cdots & (\Phi_i)_{2d_i} Q_{\varepsilon,i} \\ \vdots & \vdots & & \vdots \\ (\Phi_i)_{d_i 1} Q_{\varepsilon,i} & (\Phi_i)_{d_i 2} Q_{\varepsilon,i} & \cdots & (\Phi_i)_{d_i d_i} Q_{\varepsilon,i} \end{bmatrix}, \\
\mathcal{X}_{\mu,i} & = & \begin{bmatrix} (\Phi_i)_{11} Q_{\mu,i} & (\Phi_i)_{12} Q_{\mu,i} & \cdots & (\Phi_i)_{1d_i} Q_{\mu,i} \\ (\Phi_i)_{21} Q_{\mu,i} & (\Phi_i)_{22} Q_{\mu,i} & \cdots & (\Phi_i)_{2d_i} Q_{\mu,i} \\ \vdots & \vdots & & \vdots \\ (\Phi_i)_{d_i 1} Q_{\mu,i} & (\Phi_i)_{d_i 2} Q_{\mu,i} & \cdots & (\Phi_i)_{d_i d_i} Q_{\mu,i} \end{bmatrix}, \\
\mathcal{X}_i^k & = & \begin{bmatrix} (\Phi_i^k)_{11} N^k & (\Phi_i^k)_{12} N^k & \cdots & (\Phi_i^k)_{1d_i} N^k \\ (\Phi_i^k)_{21} N^k & (\Phi_i^k)_{22} N^k & \cdots & (\Phi_i^k)_{2d_i} N^k \\ \vdots & \vdots & & \vdots \\ (\Phi_i^k)_{d_i 1} N^k & (\Phi_i^k)_{d_i 2} N^k & \cdots & (\Phi_i^k)_{d_i d_i} N^k \end{bmatrix}, \\
\mathcal{X}_{ij} & = & \begin{bmatrix} (\Phi_{ij})_{11} N_{ij} & (\Phi_{ij})_{12} N_{ij} & \cdots & (\Phi_{ij})_{1d_j} N_{ij} \\ (\Phi_{ij})_{21} N_{ij} & (\Phi_{ij})_{22} N_{ij} & \cdots & (\Phi_{ij})_{2d_j} N_{ij} \\ \vdots & \vdots & & \vdots \\ (\Phi_{ij})_{d_i 1} N_{ij} & (\Phi_{ij})_{d_i 2} N_{ij} & \cdots & (\Phi_{ij})_{d_i d_j} N_{ij} \end{bmatrix}, \\
\mathcal{X}_{im} & = & \begin{bmatrix} (\Phi_{ij})_{11} N_{im} & (\Phi_{ij})_{12} N_{im} & \cdots & (\Phi_{ij})_{1d_i} N_{im} \\ (\Phi_{ij})_{21} N_{im} & (\Phi_{ij})_{22} N_{im} & \cdots & (\Phi_{ij})_{2d_i} N_{im} \\ \vdots & \vdots & & \vdots \\ (\Phi_{ij})_{d_i 1} N_{im} & (\Phi_{ij})_{d_i 2} N_{im} & \cdots & (\Phi_{ij})_{d_i d_i} N_{im} \end{bmatrix}, \\
\mathcal{X}_{ia}^E & = & z_i^{-1} \begin{bmatrix} (\Phi_{ij})_{11} N_{ia}^2 & (\Phi_{ij})_{12} N_{ia}^2 & \cdots & (\Phi_{ij})_{1d_i} N_{ia}^2 \\ (\Phi_{ij})_{21} N_{ia}^2 & (\Phi_{ij})_{22} N_{ia}^2 & \cdots & (\Phi_{ij})_{2d_i} N_{ia}^2 \\ \vdots & \vdots & & \vdots \\ (\Phi_{ij})_{d_i 1} N_{ia}^2 & (\Phi_{ij})_{d_i 2} N_{ia}^2 & \cdots & (\Phi_{ij})_{d_i d_i} N_{ia}^2 \end{bmatrix} \quad \text{et} \quad \mathcal{X}_{ia}^H = z_i^2 \mathcal{X}_{ia}^E,
\end{array} \right.
$$

les définitions des matrices $\mathcal{X}_{i\infty}^E$ et $\mathcal{X}_{i\infty}^H$ n'étant pas précisées ici. Si on suppose d_i identique pour toutes les cellules τ_i, toutes ces matrices exceptées $\mathcal{X}_{i\infty}^E$ et $\mathcal{X}_{i\infty}^H$ sont de taille $3d_i \times 3d_i$. Les matrices $\mathcal{X}_{i\infty}^E$ et $\mathcal{X}_{i\infty}^H$ sont de taille $3d_i \times 6d_i$. De plus, $\mathcal{X}_{\varepsilon,i}$ et $\mathcal{X}_{\mu,i}$ sont des matrices symétriques définies positives, et $\mathcal{X}_{ij}$ et $\mathcal{X}_{im}$ sont des matrices antisymétriques.

1.2.5 Mise en œuvre

On décrit ici les principaux éléments intervenant dans la mise en œuvre de la méthode GDDT en maillages simplexes non-structurés.

1.2.5.1 Les maillages simplexes

Définition 1.2.1 (Simplexes non dégénérés)
Soient $(n+1)$ points $a_j = (a_{ij})_{i=1}^n$, $1 \leq j \leq n+1$, non situés dans un même hyperplan de $\mathbb{R}^n$, c'est à dire tels que la matrice d'ordre $n+1$:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1\,n+1} \\ a_{21} & a_{22} & \dots & a_{2\,n+1} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & & a_{n\,n+1} \\ 1 & 1 & \dots & 1 \end{pmatrix},$$

est inversible. On appelle n-simplexe non dégénéré K de $\mathbb{R}^n$ l'enveloppe convexe des $(n+1)$ points $(a_j)_{j=1}^{n+1}$ de $\mathbb{R}^n$ utilisée pour former un repère affine dans un espace affine de dimension n.

Un simplexe tire son nom du fait qu'il soit l'objet géométrique le plus simple à n dimensions. Ainsi, un 1-simplexe est un segment, un 2-simplexe un triangle et un 3-simplexe un tétraèdre.

Définition 1.2.2 *Les éléments d'un simplexe sont appelés nfaces, où n est leur dimension.*

Les 0faces, 1faces, 2faces et 3faces sont appelés respectivement sommets, arêtes, faces et cellules. L'ensemble des $(n-1)$ faces d'un n-simplexe forme son enveloppe. Les $n$faces d'un simplexe sont elle-mêmes des simplexes de dimension inférieure. Par exemple, un tétraèdre aura des faces triangulaires.

1.2.5.2 Triangulation du domaine spatial

La méthode de Galerkin discontinue ne requiert pas de discrétisation particulière du domaine spatial Ω, ce dernier pouvant être une grille structurée ou non, homogène ou hybride.

Définition 1.2.3 (Triangulation)
On appelle triangulation admissible de $\bar{\Omega}$ une famille $\mathcal{T}_h = \{\tau_i\}_{1 \leq i \leq N}$, N étant le nombre total d'éléments, constituée de n-simplexes non dégénérés tels que :

- *$\tau_i \subset \Omega$ et $\bar{\Omega} = \bigcup_{i=1}^{N} \tau_i$,*
- *l'intersection $\tau_i \bigcap \tau_j$ est un m-simplexe, avec $0 \leq m \leq n-1$, dont les sommets sont des sommets de K_i et K_j, c'est à dire que :*

$$\tau_i \bigcap \tau_j = \begin{cases} \text{l'ensemble vide ou,} \\ \text{un sommet commun ou une arête commune en 2D,} \\ \text{un sommet commun, une arête et/ou une face commune en 3D.} \end{cases} \tag{1.2.10}$$

Le paramètre h est défini par :

$$h = \max diam(\tau_i) \quad avec\ diam(\tau_i) = \sup\{|x-y|;\ x, y \in \tau_i\},$$

où $|.|$ désigne la norme euclidienne. Les sommets ou nœuds de la triangulation $\mathcal{T}_h$ sont les sommets des n-simplexes τ_i.

On ne considère dans cette étude que le cas de maillages simplexes (triangles en 2D et tétraèdres en 3D).

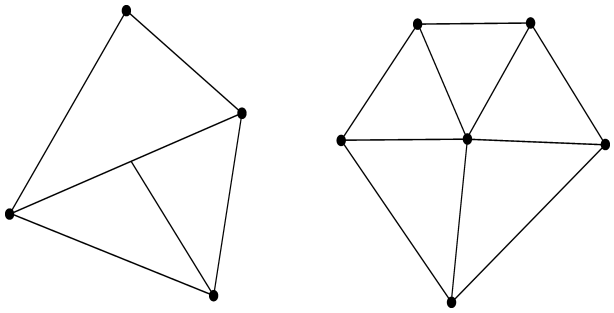


Figure 1.2 – Exemples de maillages non admissibles à gauche et admissibles à droite, en dimension 2

1.2.5.3 La maille de référence

Un maillage est généré à partir d'une *maille de référence*, qu'on note $\hat{K}$, et d'une famille de *transformations géométriques* envoyant $\hat{K}$ dans les cellules du maillage. Par la suite, on fait l'hypothèse que ces transformations sont des $\mathcal{C}^1$-difféomorphismes et pour une maille $K \in \mathcal{T}_h$, on note :

$$T_K : \hat{K} \to K,$$

la transformation géométrique correspondante.

Définition 1.2.4 *(***Maillage affine)**
Lorsque toutes les tranformations $\{T_K\}_{K\in\mathcal{T}_h}$ sont affines, c'est-à-dire lorsque pour tout $K \in \mathcal{T}_h$, il existe un vecteur $b_K \in \mathbb{R}^d$ et une matrice $J_K \in \mathbb{R}^{d,d}$ tels que :

$$T_K : \hat{K} \ni \hat{x} \to b_K + J_K\,\hat{x} \in K,$$

on dit que le maillage est affine

On se restreint dans le cas de notre étude à un maillage affine, composé de segments en dimension 1, de triangles en dimension 2 et de tétraèdres en dimension 3.

On note $\hat{T}$ l'élément de référence de sommets $\hat{S}_i$, $i = 1, \ldots, n+1$ et T un élément quelconque du maillage et on note S_i ses sommets. Un choix classique (mais non unique) pour $\hat{T}$ est de prendre comme sommets :

- Triangle
  $\hat{S}_1 = {}^t(0,0),\ \hat{S}_2 = {}^t(1,0),\ \hat{S}_3 = {}^t(0,1).$
- Tétraèdre
  $\hat{S}_1 = {}^t(0,0,0),\ \hat{S}_2 = {}^t(1,0,0),\ \hat{S}_3 = {}^t(0,1,0),\ \hat{S}_4 = {}^t(0,0,1).$

On note $\hat{F}_T$ l'application affine bijective de $\hat{T}$ dans T et F_T son inverse. Dans le cas 2D, pour un triangle T quelconque, $\hat{F}_T$ s'exprime par :

$$\hat{F}_T(\hat{X}) = \hat{A}_T\hat{X} + \hat{B}_T,$$

$$\hat{A}_T = \begin{pmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{pmatrix}, \qquad \hat{B}_T = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix},$$

où $(x_i, y_i)_{1\le i\le 3}$ sont les coordonnées des trois sommets du triangle T. L'expression de F_T est alors :

$$F_T(X) = A_T\,X + B_T,$$

$$A_T = \frac{1}{J_T}\begin{pmatrix} y_3 - y_1 & x_1 - x_3 \\ y_1 - y_2 & x_2 - x_1 \end{pmatrix}, \qquad B_T = \frac{1}{J_T}\begin{pmatrix} x_3\, y_1 - x_1\, y_3 \\ x_1\, y_2 - x_2\, y_1 \end{pmatrix},$$

où J_T est le déterminant de A vérifiant :

$$\|J_T\| = \frac{\text{aire}(T)}{\text{aire}(\hat{T})} = 2\,\text{aire}(T).$$

Les expressions pour un tétraèdre sont similaires et se calculent facilement. L'intérêt d'un élément de référence est d'éviter le calcul d'un certain nombre d'éléments sur chaque triangle (ou tétraèdre) et de les déduire simplement du triangle (ou tétraèdre) de référence via la transformation $\hat{F}$.

On a par exemple :

- Correspondance des nœuds

$$\hat{F}_T(S_j) = S_j; \quad j = 1, \ldots, n_k.$$

- Correspondance des arêtes

$$\hat{F}_T([S_i, S_j]) = [S_i, S_j]; \quad i, j = 1, 3.$$

- Correspondance des fonctions de base

$$\hat{\varphi}_j = \varphi_j \circ \hat{F}_T.$$

Cette dernière relation permet de transformer toute fonction définie sur T vers $\hat{T}$:

$$\hat{f} = f \circ \hat{F}_T.$$

1.2.5.4 Intégrales sur un élément

Pour intégrer une fonction f à valeurs réelles définie sur l'élément T, on utilise la formule de changement de variables :

$$\int_T f(\mathbf{x}) d\mathbf{x} = \|J_T\| \int_{\hat{T}} f(\hat{\mathbf{x}}) d\hat{\mathbf{x}}.$$

En pratique on utilise les relations suivantes :

- $\|J_T\| = 2\text{aire}(T)$ si T est un triangle,
- $\|J_T\| = 6\text{vol}(T)$ si T est un tétraèdre.

La formule d'intégration s'écrit donc :

$$\int_T f(\mathbf{x}) d\mathbf{x} = \frac{\text{aire}(T)}{\text{aire}(\hat{T})} \int_{\hat{T}} f(\hat{\mathbf{x}}) d\hat{\mathbf{x}}.$$

1.2.5.5 Intégrales sur une face

Le calcul des matrices de masse M^{ij} fait intervenir une intégration sur la face commune de T_i et T_j de fonctions définies sur T_i et celui de M^{ji} un produit de fonctions définies sur T_i avec des fonctions définies sur T_j. On se place ci-dessous dans le cas 2D où A_{ij} est une arête.

Calcul de M^{ij}

On rappelle l'équation de la matrice de masse (dite de surface) sur l'arête A_{ij} joignant les sommets S_i et S_j de

T_i :

$$M^{ij}_{ml} = \int_{A_{ij}} \varphi^i_l \varphi^i_m d\mathbf{s}.$$

Notons bien que les exposants dans la formule ci-dessus réfèrent aux numéros des éléments alors que les indices correspondent aux numéros des fonctions de base et ces dernières dépendent de l'élément considéré.

Le passage à l'élément de référence se fait via la formule :

$$\int_{A_{ij}} \varphi_l^i \varphi_m^i d\mathbf{s} = \frac{\|A_{ij}\|}{\|\hat{A}_{ij}\|} \int_{\hat{A}_{ij}} \hat{\varphi}_l^i \hat{\varphi}_m^i d\mathbf{s}. \tag{1.2.11}$$

Pour calculer l'intégrale sur $\hat{A}_{ij}$ de (1.2.11) on paramétrise $\hat{A}_{ij}$ pour ramener l'intégrale sur l'intervalle $[0,1]$:

$$\int_{\hat{A}_{ij}} \hat{\varphi}_l^i \hat{\varphi}_m^i d\mathbf{s} = \|\hat{A}_{ij}\| \int_0^1 \hat{\varphi}_l^i(\hat{\mathbf{x}}(t)) \hat{\varphi}_m^i(\hat{\mathbf{x}}(t)) dt.$$

L'équation (1.2.11) devient :

$$\int_{A_{ij}} \varphi_l^i \varphi_m^i d\mathbf{s} = \|A_{ij}\| \int_0^1 \hat{\varphi}_l^i(\hat{\mathbf{x}}(t)) \hat{\varphi}_m^i(\hat{\mathbf{x}}(t)) dt.$$

En pratique pour le calcul des matrices locales M_{ij}, on évalue et on stocke 3 matrices en dimension 2 (4 en dimension 3) sur $\hat{T}$ et on multiplie ces dernières par l'aire de A_{ij}. Pour certaines bases (par exemple la base des fonctions de Lagrange) une seule matrice est stockée et de plus, elle est de même dimension que la matrice de masse de même degré sur le simplexe de dimension immédiatement inférieure (1 en dimension 2 et 2 en dimension 3). Ceci vient du fait que tous les polynômes s'annulent sur cette arête à l'exception, par exemple en 2D, de $k+1$ d'entre eux pour des polynômes de degré k. Ceci n'est pas le cas pour toutes les bases et il peut être nécessaire de stocker les matrices de masse associées à toutes les arêtes de l'élément considéré.

Calcul de M^{ji}
On utilise le même procédé pour calculer les termes des matrices M_{ji} dont on rappelle la formule :

$$M_{ml}^{ji} = \int_{A_{ij}} \varphi_l^i \varphi_m^i d\mathbf{s}.$$

Il demeure le problème de la renumérotation de ces matrices de surface, les fonctions de bases étant numérotées par élément. En effet, le terme M_{11}^{ij} par exemple fait référence à la fonction de base φ_1 qui peut correspondre à un numéro $m = 1, \ldots, N_k$ dans T_i et à tout autre numéro sur T_j. Il faut donc identifier une fonction de base de numéro local m sur l'arête A_{ij} avec son numéro m_i dans T_i d'une part et son numéro m_j dans T_j d'autre part. Il y a plusieurs façons de la faire et on utilise en pratique les coordonnées des sommets de l'arête pour identifier ces nombres.

1.2.6 Discrétisation temporelle

On se place ici dans le cas 3D et on suppose, pour simplifier la présentation, que $\Gamma^a = \emptyset$ (et donc $\mathcal{F}_a = \emptyset$). Nous réécrivons les systèmes d'équations différentielles ordinaires (EDO) locaux (1.2.9) comme, $\forall \tau_i \in \mathcal{T}_h$:

$$\left\{ \begin{array}{rcl} \mathcal{X}_{\varepsilon,i} \dfrac{d\mathbf{E}_i}{dt} & = & K_i \mathbf{H}_i - \dfrac{1}{2} \displaystyle\sum_{a_{ij} \in \mathcal{F}^i} \mathcal{X}_{ij} \mathbf{H}_i - \dfrac{1}{2} \sum_{a_{ij} \in \mathcal{F}_d^i} \mathcal{X}_{ij} \mathbf{H}_j - \sum_{a_{ij} \in \mathcal{F}_m^i} \mathcal{X}_{im} \mathbf{H}_i, \\ \mathcal{X}_{\mu,i} \dfrac{d\mathbf{H}_i}{dt} & = & -K_i \mathbf{E}_i + \dfrac{1}{2} \displaystyle\sum_{a_{ij} \in \mathcal{F}^i} \mathcal{X}_{ij} \mathbf{E}_i + \dfrac{1}{2} \sum_{a_{ij} \in \mathcal{F}_d^i} \mathcal{X}_{ij} \mathbf{E}_j - \sum_{a_{ij} \in \mathcal{F}_m^i} \mathcal{X}_{im} \mathbf{E}_i. \end{array} \right. \tag{1.2.12}$$

On rappelle que $\mathbf{E}_i$ et $\mathbf{H}_i$ désignent des vecteurs $3d_i \times 1$ des degrés de liberté locaux $\mathbf{E}_{ij}$ et $\mathbf{H}_{ij}$ pour $j = 1, \ldots, d_i$ associés à la cellule τ_i. Le système (1.2.12) est équivalent à :

$$\left\{ \begin{array}{rcl} M_i^\epsilon \dfrac{d\mathbf{E}_i}{dt} & = & K_i \mathbf{H}_i - \displaystyle\sum_{k \in \mathcal{V}_i} S_{ik} \mathbf{H}_k, \\ M_i^\mu \dfrac{d\mathbf{H}_i}{dt} & = & -K_i \mathbf{E}_i + \displaystyle\sum_{k \in \mathcal{V}_i} S_{ik} \mathbf{E}_k. \end{array} \right. \tag{1.2.13}$$

Ce système d'EDO local pour chaque τ_i peut être transformé en un système d'EDO global en supposant que toutes les inconnues électriques (respectivement magnétiques) sont regroupées dans un vecteur colonne $\mathbb{E}$ (respectivement $\mathbb{H}$) de taille $d_g = \displaystyle\sum_{i=1}^{N_t} d_i$ où N_t représente le nombre d'éléments dans $\mathcal{T}_h$. Ainsi le système (1.2.13) peut s'écrire de la manière suivante :

$$\left\{ \begin{array}{rcl} \mathbb{M}^\epsilon \dfrac{d\mathbb{E}}{dt} & = & \mathbb{K}\mathbb{H} - \mathbb{A}\mathbb{H} - \mathbb{B}\mathbb{H} + \mathbb{C}_E \mathbb{E}, \\ \mathbb{M}^\mu \dfrac{d\mathbb{H}}{dt} & = & -\mathbb{K}\mathbb{E} + \mathbb{A}\mathbb{E} - \mathbb{B}\mathbb{E} + \mathbb{C}_H \mathbb{H}, \end{array} \right. \tag{1.2.14}$$

où l'on a les définitions et les propriétés suivantes :

- $\mathbb{M}^\epsilon, \mathbb{M}^\mu$ et $\mathbb{K}$ sont des matrices diagonales par bloc de taille $d_g \times d_g$ dont les blocs correspondent à M_i^ϵ, M_i^μ et K_i respectivement. $\mathbb{M}^\epsilon$ et $\mathbb{M}^\mu$ sont des matrices symétriques définies positives, et $\mathbb{K}$ est une matrice symétrique.
- $\mathbb{A}$ est aussi une matrice par bloc creuse de taille $d_g \times d_g$, dont les blocs non nuls sont égaux à S_{ik} lorsque $a_{ik} \in \mathcal{F}_h^I$. Puisque $\vec{n}_{ki} = -\vec{n}_{ik}$, on peut montrer que $(S_{ik})_{jl} = (S_{ki})_{lj}$ et alors $S_{ki} = {}^t S_{ik}$; ainsi $\mathbb{A}$ est une matrice symétrique.
- $\mathbb{B}$ est une matrice diagonale par bloc de taille $d_g \times d_g$, dont les blocs non nuls sont égaux à S_{ik} lorsque $a_{ik} \in \mathcal{F}_h^B \cap \Gamma_m$. Dans ce cas, $(S_{ik})_{jl} = -(S_{ik})_{lj}$; ainsi $\mathbb{B}$ est une matrice anti-symétrique.
- $\mathbb{C}_E$ et $\mathbb{C}_H$ sont des matrices diagonales par bloc de taille $d_g \times d_g$ associées aux termes d'intégrales de bord pour $a_{ik} \in \mathcal{F}_h^B \cap \Gamma_a$.

Par conséquent, si on pose $\mathbb{S} = \mathbb{K} - \mathbb{A} - \mathbb{B}$, le système (3.2.7) se réécrit comme :

$$\left\{ \begin{array}{rcl} \mathbb{M}^\epsilon \dfrac{d\mathbb{E}}{dt} & = & \mathbb{S}\mathbb{H} + \mathbb{C}_E \mathbb{E}, \\ \mathbb{M}^\mu \dfrac{d\mathbb{H}}{dt} & = & -{}^t\mathbb{S}\mathbb{E} + \mathbb{C}_H \mathbb{H}. \end{array} \right. \tag{1.2.15}$$

Le schéma semi-discret (3.2.8) peut être intégré en temps par l'utilisation d'un schéma leap-Frog d'ordre 2 :

$$\left\{ \begin{array}{rcl} \mathbb{M}^\epsilon \left(\dfrac{\mathbb{E}^{n+1} - \mathbb{E}^n}{\Delta t} \right) & = & \mathbb{S}\mathbb{H}^{n+\frac{1}{2}} + \mathbb{C}_E \mathbb{E}^n, \\ \mathbb{M}^\mu \left(\dfrac{\mathbb{H}^{n+\frac{3}{2}} - \mathbb{H}^{n+\frac{1}{2}}}{\Delta t} \right) & = & -{}^t\mathbb{S}\mathbb{E}^{n+1} + \mathbb{C}_H \mathbb{H}^{n+\frac{1}{2}}. \end{array} \right. \tag{1.2.16}$$

La méthode entièrement explicite DGTD-$\mathbb{P}_{p_i}$ résultante est analysée dans [Fezoui 2005] où il est montré que, lorsque $\Gamma_a = \emptyset$, la méthode est non dissipative, conserve une forme discrète de l'énergie électromagnétique et est stable sous la condition CFL suivante :

$$\Delta t \leq \frac{2}{d_2}, \quad \text{with} \quad d_2 = \| \, (\mathbb{M}^{-\mu})^{\frac{1}{2}} \; {}^t\mathbb{S} \; (\mathbb{M}^{-\epsilon})^{\frac{1}{2}} \, \|, \tag{1.2.17}$$

où $\|.\|$ désigne la norme matricielle canonique ($\forall X, \|AX\| \leq \|A\|\|X\|$), et où la matrice $(\mathbb{M}^{-\eta})^{\frac{1}{2}}$ représente l'inverse de la racine carrée de $\mathbb{M}^\eta$.

Chapitre 2

Etude numérique comparative d'interpolations polynomiales

Sommaire

2.1 Introduction

Le choix d'une méthode d'interpolation polynomiale doit prendre en compte plusieurs critères parmi lesquels le caractère nodal ou modal de l'interpolation, la nécessité de formules de quadrature pour le calcul d'intégrales élémentaires, le caractère hiérarchique ou non de l'interpolation en vue de faciliter la mise en œuvre d'une méthode où l'ordre d'approximation est variable en espace. Dans le cas des formulations Galerkin discontinues, il faut aussi tenir compte du caractère local ou non des calculs mettant en jeu un simplexe de dimension inférieure (typiquement, dans les calculs d'intégrales sur la frontière du simplexe primal). Eventuellement, on peut aussi souhaiter que l'approximation préserve des propriétés liées au modèle mathématique résolu comme la positivité de certaines quantités physiques. Un premier objectif de la présente étude est d'évaluer en détail différentes méthodes d'interpolation polynomiale en association avec la formulation GDDT-$\mathbb{P}_p$ proposée par [Fezoui 2005] et décrite au chapitre précédent. Par ailleurs, l'obtention d'une méthode GDDT-$\mathbb{P}_p$ d'ordre arbitrairement élevé nécessite l'utilisation d'un schéma d'intégration en temps de précision compatible avec l'ordre de l'approximation en espace. L'augmentation du degré d'interpolation et de l'ordre de l'intégration temporelle permet alors d'améliorer la précision et la vitesse de convergence de la méthode. Le second objectif de cette étude est donc de traiter de cette question en évaluant l'apport de schémas temporels de type saute-mouton du quatrième ordre et Runge-Kutta du quatrième ordre, comparativement au saute-mouton du second ordre adopté dans [Fezoui 2005].

Pour chacune des problématiques considérées, il existe de nombreuses options. Nous en étudions ici certaines dans le cadre de la résolution numérique des équations de Maxwell monodimensionnelles et bidimensionnelles. Toutefois, cette étude s'inscrit dans une démarche plus globale qui vise le développement d'une méthode GDDT-$\mathbb{P}_p$ d'ordre arbitrairement élevé en maillages tétraédriques pour la simulation numérique de problèmes de propagation tridimensionnels. En particulier, on cherche à concevoir une méthodologie numérique qui combine un raffinement du maillage (h-raffinement) dans les zones de moindre régularité de la solution avec un enrichissement de l'ordre d'approximation (p-enrichissement) là où la solution est régulière. Dans ce contexte, il est souhaitable d'opter pour une méthode d'interpolation qui facilite la mise en œuvre d'une stratégie p-adaptative.

La suite de ce chapitre est organisée comme suit : la section 2.2 présente quelques généralités utiles à l'analyse des fonctions de bases polynomiales dans un n-simplexe non dégénéré de $\mathbb{R}^n$, et fournit un cadre préliminaire à l'étude. La section 2.3 passe en revue différentes méthodes d'interpolation particulièrement adaptées aux triangulations par des simplexes, utilisées depuis longtemps dans les méthodes d'éléments finis continues, les méthodes spectrales ou plus récemment dans les méthodes Galerkin discontinues. La section 2.4 traite de la résolution numérique des équations de Maxwell en dimension 1 et décrit la formulation Galerkin discontinue correspondante ainsi que les différents schémas en temps considérés. Plusieurs résultats numériques sont présentés. Une étude similaire appliquée à la résolution numérique des équations de Maxwell en dimension 2 est proposée dans la section 2.6. Enfin, la section 2.7 conclue cette étude.

2.2 Généralités sur les fonctions de bases polynomiales

2.2.1 Interpolation polynomiale dans une méthode Galerkin discontinue

La formulation Galerkin discontinue décrite à la section 1.2 du chapitre précédent repose sur l'introduction d'un ensemble de fonctions de base locales $(\varphi_{ij})_{1\leq j\leq d_i}$ pour chaque élément $(\tau_i)_{1\leq i\leq N}$ du maillage

où $d_i = p_i + 1$ représente le nombre de degrés de liberté caractérisant l'approximation de degré p_i des composantes du champ électromagnétique dans l'élément $(\tau_i)_{1\leq i\leq N}$. Dans chaque élément, le champ $\mathbf{W}_i(\mathbf{x}_i)$ est approché par une combinaison linéaire de fonctions de base locales polynomiales, supposées linéairement indépendantes engendrant l'espace des fonctions polynomiales de degré au plus p_i sur τ_i, noté $\mathcal{P}_i = \mathbb{P}_{p_i}[\tau_i]$:

$$\mathbf{W}_i(\mathbf{x}_i) = \sum_{j=0}^{p_i} \mathbf{W}_{ij}\, \varphi_{ij}(\mathbf{x}_i). \tag{2.2.1}$$

Nous nous restreignons ici pour simplifier au cas de maillages conformes avec des bases de même degré dans tous les éléments du maillage (*i.e.* $p_i = p$ pour $1 \leq i \leq N$). On peut ainsi représenter la solution globale comme la somme directe de solutions polynomiales locales :

$$\mathbf{W}(\mathbf{x}) \simeq \mathbf{W}^h(\mathbf{x}) = \bigoplus_{i=1}^{N} \mathbf{W}_i^h(\mathbf{x}_i). \tag{2.2.2}$$

Pour alléger la notation dans la suite, on abandonnera l'indice h caractérisant la solution approchée.

Il y a 3 différentes approches dans l'implémentation des méthodes Galerkin discontinues, à savoir, la version h, la version p, et la version hp. De façon similaire aux méthodes d'éléments finis, la version h (où h représente une taille caractéristique du pas de discrétisation) autorise la taille des éléments à diminuer pour atteindre la convergence mais impose à la base polynomiale d'être de degré fixe au sein de chaque élément. On assure ainsi une flexibilité dans la manipulation de complexités géométriques en permettant des raffinements locaux. Dans la version p alternative, un maillage fixe est utilisé et la convergence est obtenue en augmentant le degré du polynôme au sein de chaque élément. Une approximation de type p présente l'avantage d'accélérer la convergence dans le cas de problèmes réguliers. Si le domaine de calcul entier est traité comme un élément unique, la méthode de type p s'assimile alors à une méthode spectrale. Notons qu'à l'exception des méthodes spectrales globales, la plupart des méthodes de type p requièrent nécessairement une décomposition de type h pour générer le maillage initial sur lequel l'approximation de type p est appliquée. Enfin la version hybride hp combine h-raffinement et p-enrichissement locaux et incorpore à la fois les méthodes spectrales multi-domaines et les méthodes d'éléments finis d'ordre élevé. On parle alors de méthode hp-adaptative.

Une composante importante dans l'implémentation d'une méthode Galerkin discontinue est le choix des fonctions de base. Typiquement, en une dimension d'espace, les différentes bases seront exprimées sur l'élément de référence $\tau_{st} = [0, 1]$ et on utilisera une transformation inversible faisant correspondre les nœuds de la maille de référence à ceux de la maille réelle $(\tau_i)_{1\leq i\leq N}$, de manière à faciliter le calcul d'intégrales et de différentiations pour l'évaluation des matrices élementaires intervenant dans la formulation variationnelle du problème. On parle alors d'approximation géométrique car les nœuds après transformation et ceux de l'élément réel sont confondus, mais les autres points sont distincts. Souvent, la transformation en question est de type affine, mais on doit faire appel à des transformations d'ordre supérieur lorsqu'il s'agit de prendre en compte des formes courbes. On pourra alors appliquer sur chaque sous-domaine élémentaire $(\tau_i)_{1\leq i\leq N}$ une approximation polynomiale d'ordre arbitraire et ainsi agir localement sur le nombre de degrés de liberté pour obtenir la précision désirée.

Des formulations Galerkin discontinues basées sur une interpolation nodale d'ordre élevé pour la résolution des équations de Maxwell du premier ordre en domaine temporel utilisant un degré polynomial variant localement ont été étudiées dans [Fahs 2008]. Il a été montré que la méthode GDDT-$\mathbb{P}_{p_i}$:$\mathbb{P}_{p_j}$ résultante de type hp présente plusieurs avantages en comparaison avec une méthode de h-raffinement classique en permettant notamment de réduire considérablement les coûts en temps de calcul et consommation mémoire. Pour une méthode p-adaptative, il semble préférable d'utiliser des fonctions de base hiérarchiques orthogonales [Appell 1926] - [Dubiner 1991] - [Farouki 2003] - [Koorwinder 1975] - [Owens 1998] - [Proriol 1957] - [Xu 2001] où l'incrément d'un ordre repose sur l'ajout de fonctions de base (*i.e.* l'ensemble des fonctions d'ordre p est inclu dans l'ensemble des fonctions d'ordre $p+1$). L'intérêt premier de

ces méthodes tient au fait qu'il est possible de varier l'ordre des fonctions sur les éléments du maillage pour contrôler la distribution des degrés de liberté tout en préservant la conformité de l'espace d'approximation. Dans ce qui suit on se propose d'évaluer différentes méthodes d'interpolation polynomiale pour en faire ressortir plus distinctement leurs forces et faiblesses en vue de l'élaboration d'une méthode Galerin discontinue *hp*-adaptative.

2.2.2 Les fonctions barycentriques

Définition 2.2.1 (Coordonnées barycentriques)
Dans un n-simplexe non dégénéré K de $\mathbb{R}^n$, on définit les $(n+1)$ coordonnées barycentriques $(\lambda_1, \ldots, \lambda_{n+1})$ telles que pour tout $i \in \{1, \ldots, n+1\}$:

$$\lambda_i : \mathbb{R}^n \ni x \to 1 - \frac{(x - s_i).n_i}{(s_j - s_i).n_i} \in \mathbb{R},$$

où s_j est un des sommets de K situés sur la face F_i opposée à s_i, $(x - s_i)$ le vecteur reliant s_i à x et $(s_j - s_i)$ le vecteur reliant s_i à s_j.

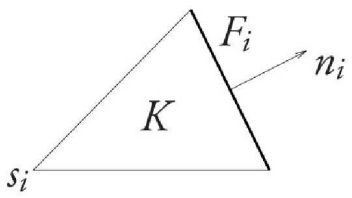


Figure 2.1 – Un triangle K, un sommet s_i, la face opposée F_i et la normale extérieure n_i

La définition 2.2.1 de λ_i est clairement indépendante du choix particulier du sommet s_j sur la face F_i. La coordonnée barycentrique λ_i est une fonction affine qui vaut 1 en s_i et s'annule sur la face F_i. De plus, ses courbes de niveau sont des hyperplans (des droites si on est en dimension 2) qui sont parallèles à la face F_i. Le barycentre de K a toutes ses coordonnées barycentriques égales à $\dfrac{1}{n+1}$.

Si K est le simplexe unité, on a :
- en dimension 1, $\lambda_0 = 1 - x_1, \quad \lambda_1 = x_1$;
- en dimension 2, $\lambda_0 = 1 - x_1 - x_2, \quad \lambda_1 = x_1 \quad$ et $\quad \lambda_2 = x_2$;
- en dimension 3, $\lambda_0 = 1 - x_1 - x_2 - x_3, \quad \lambda_1 = x_1, \quad \lambda_2 = x_2 \quad$ et $\quad \lambda_3 = x_3$.

Définition 2.2.2 *Soit K un n-simplexe non dégénéré de $\mathbb{R}^n$ de sommets $(a_j)_{1 \le j \le n+1}$ et $x \in \mathbb{R}^n$. Alors x est caractérisé par ses coordonnées barycentriques $\lambda_j^K(x) \in \mathbb{R}$, où $1 \le j \le n+1$, par rapport à K, définies comme solutions du système linéaire :*

$$\sum_{j=1}^{n+1} \lambda_j^K = 1 \quad et \quad x = \sum_{j=1}^{n+1} \lambda_j^K(x) a_j.$$

Remarque : Le système défini par 2.2.2 s'écrit encore $A\lambda = X$, où $\lambda \in \mathbb{R}^{n+1}$ a pour coefficients les $\lambda_j^K(x), X = {}^t(1, x_1, \ldots, x_n) \in \mathbb{R}^{n+1}$ et A est donnée par 1.2.1. L'inversibilité de A assure l'existence et l'unicité de λ donc des coordonnées barycentriques $\lambda_j^K(x)$.

Les coordonnées barycentriques permettent de donner une caractérisation simple du n-simplexe K :

$$K = \{x \in \mathbb{R}^n | \quad 0 \leq \lambda_i^K \leq 1, \quad \forall i = 1, \ldots, n+1\}. \tag{2.2.3}$$

De plus, on remarque que l'on a $\lambda_i^K(a_j) = \delta_{ij}$. En conséquence on a :

- $\lambda_k^K = 0$ est l'équation de l'arête $[S_i S_j]$ avec i et j différents de k et ceci est vrai dans tout triangle et plus généralement dans tout n-simplexe.
- La propriété 2.2.3 ci-dessus est utilisée pour savoir si un point de $\mathbb{R}^n$ appartient au n-simplexe.

2.2.3 Les treillis

Définition 2.2.3 (Treillis)
Soit K un n-simplexe non dégénéré de $\mathbb{R}^n$ de coordonnées barycentriques associées $\lambda_j^K \in \mathbb{R}$, où $1 \leq j \leq n+1$. On appelle treillis d'ordre $k \in \mathbb{N}^*$ de K l'ensemble :

$$\Sigma_k = \{x \in K | \quad \lambda_j(x) \in \{0, \frac{1}{k}, \ldots, \frac{k-1}{k}, 1\}, \quad \forall j = 1, \ldots, n+1\}.$$

Remarque : On définit Σ_0 comme étant le singleton réduit au barycentre de K.

Exemples :

- Pour $k = 1$, on a $\Sigma_1 = \{x \in K | \quad \lambda_j(x) \in \{0, 1\}, \quad \forall j = 1, \ldots, d+1\}$ donc Σ_1 est composé des sommets de K.
- Pour $k = 2$, Σ_2 est l'ensemble des sommets et des points milieux (*i.e.* les centres des arêtes). Le treillis d'ordre 2 est représenté sur la figure 2.2 pour un triangle et un tétraèdre.
- De la même manière, le treillis d'ordre 3 est représenté sur la figure 2.3 pour un triangle et un tétraèdre.

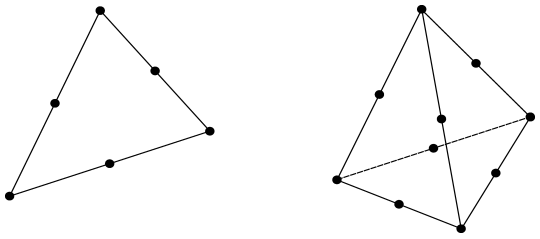


Figure 2.2 – Treillis d'ordre 2 pour un triangle à gauche et pour un tétraèdre à droite

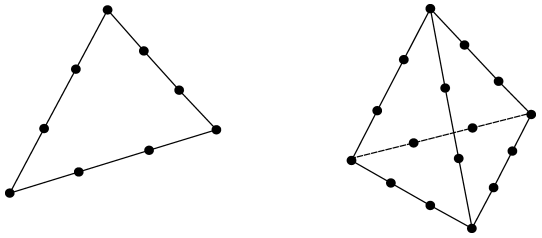


Figure 2.3 – Treillis d'ordre 3 pour un triangle à gauche et pour un tétraèdre à droite

Dimension de Σ_k : La dimension n_k de l'ensemble Σ_k est donnée par :

$$n_k = \text{Card}(\Sigma_k) = C_{k+n}^k = \frac{(k+1)\ldots(k+n)}{n!}.$$

Notation : On note $\mathbb{P}_p$ l'espace vectoriel des polynômes à coefficients réels de $\mathbb{R}^n$ de degré inférieur ou égal à p, *i.e.* tout polynôme p de $\mathbb{P}_p$ est de la forme :

$$\forall x \in \mathbb{R}^n, \quad p(x) = \sum_{\substack{0 \leq i_1, \ldots, i_n \leq p \\ i_1 + \ldots + i_n \leq p}} \alpha_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n} \quad \text{où} \quad \alpha_{i_1,\ldots,i_n} \in \mathbb{R}.$$

Remarque : On peut montrer que l'on a : $\mathrm{Card}(\Sigma_p) = \dim(\mathbb{P}_p)$.

Lemme 2.2.1 *Soient K un n-simplexe non dégénéré de $\mathbb{R}^n$ et, pour $k \in \mathbb{N}^*$, Σ_k le treillis d'ordre k de K. On désigne par $(\sigma_j)_{1\leq j\leq N_k}$ les points de Σ_k. Alors, tout polynôme de $\mathbb{P}_k$ est déterminé de manière unique par ses valeurs aux points $(\sigma_j)_{1\leq j\leq N_k}$. Plus précisemment, il existe une base $(\psi_j)_{1\leq j\leq N_k}$ de $\mathbb{P}_k$ telle que :*

$$\psi_j(\sigma_i) = \delta_{ij} \quad \textit{pour tout} \quad 1 \leq i, j \leq N_k.$$

Nous verrons par la suite que ce lemme a une application importante puisqu'il permet de construire une base de l'espace d'approximation. En effet, les points équidistants $\sigma_j, \quad j = 1, \ldots, N_k$ de Σ_k peuvent ainsi définir les nœuds de l'interpolation de Lagrange de degré k. Toutefois ce choix des points n'est pas optimal en particulier lorsque le phénomène de Runge se manifeste, comme nous le détaillerons par la suite. Pour l'interpolation de certaines fonctions, l'erreur d'interpolation peut en effet beaucoup varier d'une fonction à l'autre et peut s'avérer très grande au point de créer une instabilité de la méthode numérique. Des oscillations peuvent même apparaître au bord d'un intervalle d'interpolation, l'amplitude de ces oscillations augmentant avec le degré du polynôme d'interpolation.

2.3 Classification des bases

Lorsque l'on utilise des éléments finis de degré élevé, il est important de bien choisir les fonctions de base. Plusieurs facteurs rentrent en jeu dans le choix d'une représentation polynomiale pour le champ $\mathbf{W}_i(\mathbf{x})$:

- L'indépendance linéaire de la base choisie.
- La possibilité d'imposer facilement la valeur de l'interpolé sur la frontière de K. Cette propriété est essentielle lorsqu'on souhaite raccorder de façon continue des interpolés sur des mailles adjacentes, mais présente peu d'intérêt lorsqu'on utilise une formulation Galerkin discontinue.
- La possibilité d'inverser facilement la matrice de masse élémentaire $\Phi_{ij} \in \mathbb{R}^{p,p}$ dont les composantes sont

$$\Phi_{ij} = \int_K \varphi_i \varphi_j dx \quad i, j \in \{1, \ldots, p\}.$$

Cette section présente quelques bases d'interpolation polynomiale permettant de satisfaire (au moins en partie) les critères ci-dessus pour des éléments finis de degré élevé. On abordera successivement les bases modales, où les fonctions sont définies directement sans la médiation de degrés de liberté, puis les bases nodales, qui sont associées à un ensemble de points et se construisent généralement à partir des polynômes de Lagrange et enfin les bases hiérarchiques particulièrement adaptées aux stratégies de type *p*- et *hp*-adaptatives. Il est important de marquer la distinction entre une représentation nodale basée sur une interpolation polynomiale classique et une méthode de collocation dans laquelle la solution satisfait exactement l'équation à résoudre aux points nodaux. Dans le cas des bases modales, les polynômes de base sont déterminés sans rapport aucun avec un ensemble de points connu et il faut alors des méthodes de projection voire d'interpolation-projection pour calculer les coefficients des fonctions dans cette base. Signalons que ces bases modales sont l'ingrédient principal des méthodes justement appelées *spectrales*. La présentation est restreinte dans un premier temps aux éléments finis en dimension 1 et on pose $\tau_{st} = K = [0, 1]$.

2.3.1 Bases modales

L'exemple le plus connu est la famille des polynômes de Jacobi dont ceux de Legendre et ceux de Tchebychev sont les plus utilisés. Ces polynômes, qui sont par ailleurs orthogonaux, sont intrinsèquement définis sur $\mathbb{R}$, leur extension aux dimensions supérieures peut se faire par un simple produit tensoriel ce qui restreint cette extension aux maillages de type cartésien (rectangles en dimension deux et héxaèdres droits en dimension trois). De nombreuses extensions au cas des éléments triangulaires ou tétraédriques sont néanmoins proposées dans la littérature conduisant généralement à la perte des bonnes propriétés de ces bases en maillage cartésien (orthogonalité par exemple).

2.3.1.1 Fonctions de base de Legendre

Définition 2.3.1 (Polynômes de Legendre)
Les polynômes de Legendre $\{G_m\}_{m\geq 0}$ sur l'intervalle de référence $\tau_{st} = [0,1]$ sont définis par la relation différentielle :

$$G_m(t) = \frac{1}{m!}\frac{d^m}{dt^m}(t^2 - t)^m.$$

A l'origine, les polynômes de Legendre sont un exemple très classique d'une famille de polynômes issus du procédé d'orthogonalisation de Gram-Schmidt à partir des polynômes de la base canonique pour le produit scalaire : $(P|Q) = \int_0^1 P(t)Q(t)\,dt$. On peut, grâce aux polynômes de Legendre, obtenir une formule de quadrature sur m points qui est exacte pour les polynômes de degré $\leq 2m-1$ en prenant comme points d'intégration les zéros du polynôme de Legendre G_m. Ce polynôme est de degré m, il vérifie $G_m(0) = (-1)^m$, $G_m(1) = 1$ et ses m racines se trouvent toutes dans K. Dans la littérature, les polynômes de Legendre sont plus fréquemment définis sur l'intervalle de référence $K = [-1,1]$. En notant $G_m^*(s)$ les polynômes de Legendre définis sur $[-1,1]$, on a $G_m^*(s) = G_m(\frac{1}{2}(1+s))$. La même remarque s'applique pour les polynômes de Jacobi définis par la suite.

La propriété fondamentale satisfaite par les polynômes de Legendre est qu'ils forment une famille de polynômes orthogonaux sur l'intervalle $[0,1]$. On a pour tout $m, n \geq 0$ $(m \neq n)$:

$$\int_0^1 G_m(t)\,G_n(t)\,dt = \frac{1}{2m+1}\,\delta_{mn}.$$

Par conséquent, la matrice de masse élémentaire associée à la base $\{G_0, \ldots, G_k\}$ est diagonale et son nombre de conditionnement vaut $(2k+1)$.

Une deuxième propriété intéressante des polynômes de Legendre est qu'ils forment une base hiérarchique de $\mathbb{P}_k$ au sens de la définition suivante.

Définition 2.3.2 (Base hiérarchique)
Soit un entier $k \geq 0$. On dit que la famille de polynômes $\{\mathcal{P}_0, \ldots, \mathcal{P}_k\}$ forme une base hiérarchique de $\mathbb{P}_k$ si pour tout $l \in \{0, \ldots, k\}$, la famille $\{\mathcal{P}_0, \ldots, \mathcal{P}_l\}$ forme une base de $\mathbb{P}_l$.

L'intérêt des bases hiérarchiques est que, si l'on souhaite augmenter le degré d'interpolation localement sans raffiner globalement le maillage (par exemple pour améliorer la précision de l'interpolé), il suffit de rajouter des nouveaux polynômes à la base sans devoir modifier les anciennes fonctions de base. L'exemple le plus simple de base hiérarchique de $\mathbb{P}_k$, en toute dimension et ce pour tout ouvert de $\mathbb{R}^n$, est la base canonique définie pour tout $n \geq 1$ et pour tout $k \geq 0$ par :

$$\alpha_i \in \mathbb{N}, \quad \|\alpha\| = \sum_{i=1}^{n} \alpha_i = k, \quad P_\alpha(x_1, \ldots, x_n) = x_1^{\alpha_1} \ldots x_2^{\alpha_n},$$

que nous étudierons plus loin. Généralement, le caractère hiérarchique d'une base reste valide dans un ouvert de $\mathbb{R}^n$ pour $n > 1$ si cet ouvert est un produit cartésien d'intervalles de $\mathbb{R}$ et la préservation de cette propriété dans un triangle est un sujet difficile que nous aborderons plus loin.

Les polynômes de Legendre appartiennent à la famille des polynômes de Jacobi, communément utilisés dans les stratégies de type p-adaptative. :

$$\left\{\mathcal{J}_m^{\alpha\beta}, \quad m = 0, \cdots, p\right\},$$

solutions du problème de Sturm-Liouville singulier suivant :

$$\frac{d}{dx}(1-x^2)w(x)\frac{d}{dx}\mathcal{J}_m^{\alpha\beta}(x) + m(m+\alpha+\beta+1)w(x)\mathcal{J}_p^{\alpha\beta}(x) = 0, \quad x \in [-1,1], \tag{2.3.1}$$

où la fonction de poids est égale à $w(x) = (1-x)^\alpha(1+x)^\beta$, avec $\alpha, \beta > -1$.

Définition 2.3.3 (Polynômes de Jacobi)
Soient deux entiers $\alpha > -1$ et $\beta > -1$. Les polynômes de Jacobi $\{\mathcal{J}_m^{\alpha\beta}\}_{m\geq 0}$ sur l'intervalle de référence $\tau_{st} = [0,1]$ s'expriment sous la forme :

$$\mathcal{J}_m^{\alpha\beta}(t) = \frac{(-1)^m}{m!}\,2^{-\alpha-\beta}(1-t)^{-\alpha}t^{-\beta}\frac{d^m}{dt^m}\left((1-t)^{\alpha+m}t^{\beta+m}\right).$$

L'orthogonalité pondérée de ces polynômes est une conséquence directe de l'équation (2.3.1). Les polynômes de Jacobi sont tels que pour tout $m, n \geq 0$:

$$\int_0^1 (1-t)^\alpha t^\beta \mathcal{J}_m^{\alpha\beta}(t)\mathcal{J}_n^{\alpha\beta}(t)dt = c_{m,\alpha,\beta}\delta_{mn},$$

avec $c_{m,\alpha,\beta} = \dfrac{1}{2m+\alpha+\beta+1}\dfrac{(m+\alpha)!(m+\beta)!}{m!(m+\alpha+\beta)!}$.

Les polynômes de Legendre correspondent au cas particulier où $\alpha = \beta = 0$, pour tout $m \geq 0$. Pour plus d'informations sur les polynômes de Legendre et de Jacobi, on pourra consulter les travaux d'Abramowitz et Stegun [Abramowitz 1964] ou de Karniadakis et Spencer [Karniadakis 2005].

Le premier polynôme $\varphi_0(x)$ (d'ordre 0) est la constante 1, puis les polynômes de degrés supérieurs $\varphi_p(x)$ sont définis par récurrence :

$$(p+1)\,\varphi_{p+1}(x) = (2p+1)(2x-1)\,\varphi_p(x) - p\,\varphi_{p-1}(x).$$

Ainsi, l'expression analytique des six premières fonctions de base de Legendre est la suivante :

$$\begin{cases} \varphi^0(x) &= 1, \\ \varphi^1(x) &= 2x-1, \\ \varphi^2(x) &= 6x^2-6x+1, \\ \varphi^3(x) &= 20x^3-30x^2+12x-1, \\ \varphi^4(x) &= 70x^4-140x^3+90x^2-20x+1, \\ \varphi^5(x) &= 252x^5-630x^4+560x^3-210x^2+30x-1. \end{cases} \tag{2.3.2}$$

On visualise sur la figure 2.4 les fonctions de base de Legendre sur l'élément de référence $\tau_{st} = [0,1]$ pour les interpolations $\mathbb{P}_1$ à $\mathbb{P}_5$.

En 1885, Stieltjes découvrit que les zéros des polynômes de Jacobi peuvent être interprétés comme des positions d'équilibre d'un problème d'électrostatique faisant intervenir un nombre fini de charges. En conséquence, il fut établit que les points de quadrature de Gauss des polynômes orthogonaux classiques peuvent être déterminés à partir de la distribution stationnaire des charges minimisant l'énergie pour un problème d'électrostatique.

Les polynômes orthogonaux jouent un rôle essentiel dans la conception de fonctions de base d'ordre élevé optimales et particulièrement dans la construction de formules de quadratures ayant un degré d'exactitude maximal. L'idée d'une quadrature de Gauss classique consiste à approcher la valeur numérique d'une intégrale par une somme pondérée prise en un certain nombre de points du domaine d'intégration :

$$\int_a^b f(x)w(x)dx = \sum_{i=0}^{p-1} f(x_i)w_i + E(f), \tag{2.3.3}$$

où x_i représentent les noeuds de quadrature, w_i les coefficients de quadrature ou poids et $E(f)$ l'erreur entre l'intégrale exacte et son approximation. Le domaine d'intégration $[a, b]$ et la fonction de pondération $w(x)$ déterminent le type de la quadrature de Gauss. Communément, pour une quadrature de Legendre, la fonction de pondération $w(x)$ est prise égale à 1 et l'intervalle $[a, b]$ a pour valeur $[-1, 1]$. Les nœuds sont alors déterminés comme les p racines du p-ème polynôme orthogonal associé à la formule de quadrature (polynômes de Legendre pour la formule de *Gauss-Legendre*, etc.). On distingue trois différents types de quadratures de Gauss associées aux polynômes de Legendre connues sous le nom de *Gauss-Legendre*, *Gauss-Radau-Legendre* et *Gauss-Lobatto-Legendre*. Dans toutes les quadratures de Gauss, les nœuds sont pris dans le domaine d'intégration mais pour une formule de *Gauss-Radau-Legendre*, on inclue également une borne de l'intervalle voire même les deux dans le cas d'une quadrature de type *Gauss-Lobatto-Legendre*. On a alors le résultat suivant : une formule à p points admet un degré d'exactitude de $2p - 1$ pour une quadrature de type *Gauss-Legendre*, de $2p - 2$ pour une quadrature de type *Gauss-Radau-Legendre* et enfin de $2p - 3$ pour une quadrature de type *Gauss-Lobatto-Legendre*.

2.3.1.2 Fonctions de base canoniques

Comme nous l'avons vu précédemment, l'exemple le plus simple de base hiérarchique en toute dimension et pour tout ouvert de $\mathbb{R}^n$ reste la base canonique définie pour tout $n \geq 1$ et pour tout $k \geq 0$ par :

$$\alpha_i \in \mathbb{N}, \quad \|\alpha\| = \sum_{i=1}^{n} \alpha_i = k, \quad P_\alpha(x_1, \ldots, x_n) = x_1^{\alpha_1} \ldots x_2^{\alpha_n}.$$

Etant donnés $p + 1$ nœuds distincts $(x_k)_{0 \leq k \leq p}$ et $p + 1$ valeurs associées $(y_k)_{0 \leq k \leq p}$, on considère le problème d'interpolation suivant :

$$\text{Trouver } \mathbf{W}_i \in \mathbb{P}_p[\tau_i] \text{ tel que } \mathbf{W}_i(x_k) = y_k, \quad k = 0, \cdots, p.$$

La base canonique est la base modale qui paraît la plus naturelle pour travailler dans l'espace vectoriel étudié. Le polynôme d'interpolation $\mathbf{W}_i$ solution du problème précédent, exprimé dans la base canonique de $\mathbb{P}_p[\tau_i]$, s'écrit :

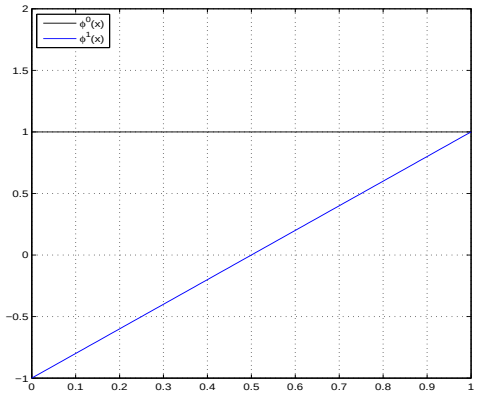
$$\mathbf{W}_i(x) = \sum_{k=0}^{p} \mathbf{W}_{ij}\, x^j.$$

Par conséquent, les coefficients $\mathbf{W}_{ij}$ doivent vérifier :

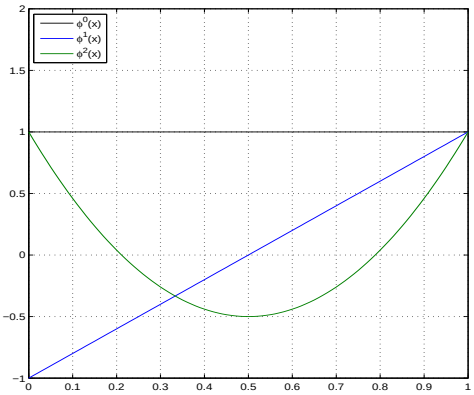
$$\sum_{j=0}^{p} \mathbf{W}_{ij}\, x_k^j = y_k, \quad k = 0, \cdots, p.$$

Autrement dit, les $\mathbf{W}_{ij}$ doivent être calculés comme solution du système linéaire :

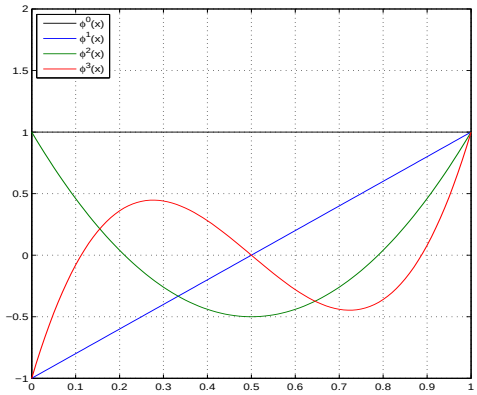
$$\begin{pmatrix} 1 & x_0^1 & \cdots & x_0^p \\ 1 & x_1^1 & \cdots & x_1^p \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_p^1 & \cdots & x_p^p \end{pmatrix} \begin{pmatrix} \mathbf{W}_{i0} \\ \mathbf{W}_{i1} \\ \vdots \\ \mathbf{W}_{ip} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_p \end{pmatrix}.$$



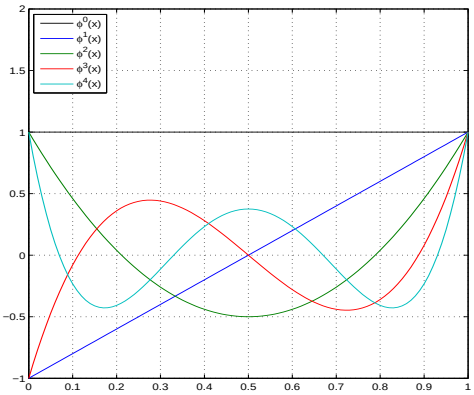
Interpolation $\mathbb{P}_1[0,1]$



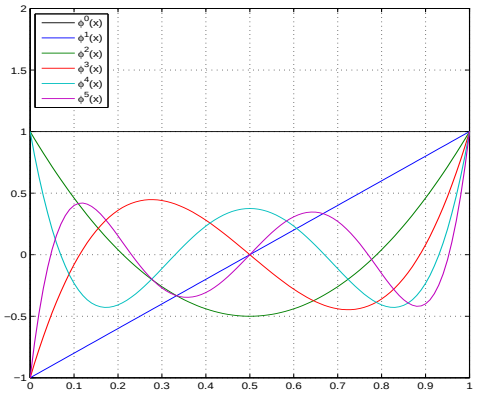
Interpolation $\mathbb{P}_2[0,1]$



Interpolation $\mathbb{P}_3[0,1]$



Interpolation $\mathbb{P}_4[0,1]$



Interpolation $\mathbb{P}_5[0,1]$

Figure 2.4 – Fonctions de base de Legendre sur l'élément de référence $\tau_{st} = [0,1]$

La matrice de ce système est la *matrice de Vandermonde* qui établit la connection entre les modes, $\mathbf{W}_{ik}$, et les valeurs nodales, $(\mathbf{W}_i(x_k))_{0\leq k\leq p}$. Son déterminant vaut $\det(V) = \prod_{i<j}(x_j - x_i)$. Si les $(x_k)_{0\leq k\leq p}$ sont distincts deux à deux, le déterminant de Vandermonde est donc non nul, et le système admet une solution unique. Si l'on exprime le polynôme d'interpolation dans la base canonique, on doit résoudre un système linéaire de $p+1$ équations à $p+1$ inconnues. Sous cette forme, le calcul, qui nécessite d'inverser la *matrice de Vandermonde*, est donc coûteux ($\mathcal{O}(p^3)$ opérations). De plus, la matrice étant mal conditionnée, la solution de ce système se montre donc très sensible aux erreurs d'arrondis.

On visualise sur la figure 2.5 les fonctions de base canoniques sur l'élément de référence pour les interpolations $\mathbb{P}_1$ à $\mathbb{P}_5$.

2.3.1.3 Fonctions de base de type Taylor

La formule de Taylor permet l'approximation d'une fonction plusieurs fois dérivable au voisinage d'un point par un polynôme dont les coefficients dépendent uniquement des dérivées de la fonction en ce point. En fait il existe trois versions de la formule de Taylor de nature très différentes : la formule de Taylor-Young, la formule de Taylor-Lagrange et la formule de Taylor avec reste intégral, énoncées de la moins précise à la plus précise. Notons que les hypothèses nécessaires d'une formule à l'autre sont conformément de plus en plus fortes.

La formule de Taylor-Young est une formule locale, qui donne des informations au voisinage d'un point. C'est elle notamment qui donne l'existence de développements limités et qui sert pour faire des études locales de courbes. Supposons que f soit à valeurs réelles de classe C^p sur un intervalle I de $\mathbb{R}$. Soit x_0 un point intérieur à I, alors pour tout $h \in \mathbb{R}$ tel que $x_0 + h$ appartienne à I, on peut écrire :

$$f(x_0 + h) = \sum_{k=0}^{p} \frac{h^k}{k!} f^{(k)}(x_0) + h^p \varepsilon(h), \tag{2.3.4}$$

où $\varepsilon(h)$ est une fonction qui tend vers 0 quand h tend vers zéro. La somme finie de l'équation (2.3.4) s'appelle le polynôme de Taylor de f à l'ordre p au point x_0. Taylor ne s'est pas vraiment préoccupé de la forme du reste, il faut attendre ses successeurs pour voir se développer une maîtrise du reste dans certaines conditions plus précises.

La formule de Taylor-Lagrange est bien plus puissante que la précédente car elle permet une étude plus globale et une majoration effective du reste. En effet, Supposons que f soit de classe C^{p+1} sur I. Alors, pour tout $h \in \mathbb{R}$ tel que $x_0 + h$ appartienne à I, il existe $\theta \in]0, 1[$ tel que l'on ait :

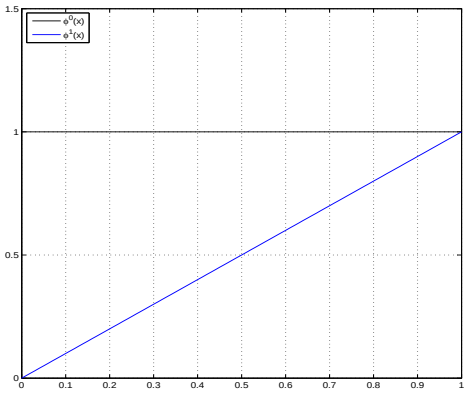
$$f(x_0 + h) = \sum_{k=0}^{p} \frac{h^k}{k!} f^{(k)}(x_0) + \frac{h^{p+1}}{(p+1)!} f^{(p+1)}(x_0 + \theta h). \tag{2.3.5}$$

Finalement la formule de Taylor avec reste intégral, est la seule à donner une expression précise du reste. Elle est très utile notamment lorsqu'on s'intéresse à la régularité de ce reste. Supposons que f soit de classe C^{p+1} sur I. Alors, pour tout $h \in \mathbb{R}$ tel que $x_0 + h$ appartienne à I, on a :

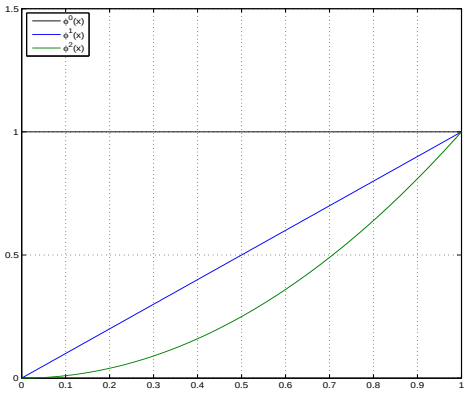
$$f(x_0 + h) = \sum_{k=0}^{p} \frac{h^k}{k!} f^{(k)}(x_0) + \frac{h^{p+1}}{p!} \int_0^1 (1-t)^p f^{(p+1)}(x_0 + th) dt. \tag{2.3.6}$$

Pour certaines fonctions f, on peut montrer que le reste tend vers zéro quand p tend vers l'infini ; ces fonctions peuvent être développées en *séries de Taylor* dans un voisinage du point x_0 et sont appelées des *fonctions analytiques*.

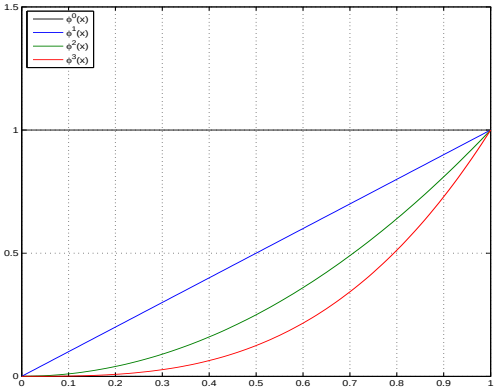
Le champ $\mathbf{W}$ sera alors approché par un développement en série de Taylor au centre de l'élément de référence $\tau_{st} = [0, 1]$, ce qui pourra s'exprimer comme une combinaison de valeurs moyennes sur l'élément et de dérivées calculées au centre de l'élément. Ainsi les inconnues à déterminer dans cette formulation sont des variables moyennées sur l'élément ainsi que leurs dérivées au centre de l'élément. Si



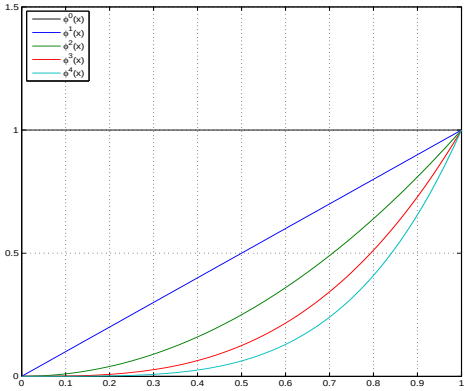
Interpolation $\mathbb{P}_1[0,1]$



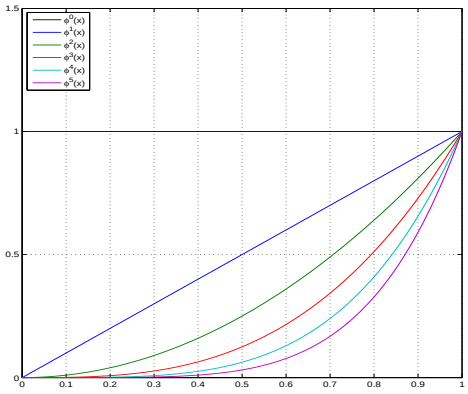
Interpolation $\mathbb{P}_2[0,1]$



Interpolation $\mathbb{P}_3[0,1]$



Interpolation $\mathbb{P}_4[0,1]$



Interpolation $\mathbb{P}_5[0,1]$

Figure 2.5 – Fonctions de base canoniques sur l'élément $\tau_{st} = [0,1]$

nous développons la solution polynomiale $\mathbf{W}^h$ en une série de Taylor tronquée à l'ordre six au centre de l'élément de référence, il vient :

$$\begin{aligned}\mathbf{W}^h = \mathbf{W}(0.5) + \left.\frac{d\mathbf{W}}{dx}\right|_{0.5}(x-0.5) + \left.\frac{d^2\mathbf{W}}{dx^2}\right|_{0.5}\frac{(x-0.5)^2}{2} \\ + \left.\frac{d^3\mathbf{W}}{dx^3}\right|_{0.5}\frac{(x-0.5)^3}{3!} + \left.\frac{d^4\mathbf{W}}{dx^4}\right|_{0.5}\frac{(x-0.5)^4}{4!} + \left.\frac{d^5\mathbf{W}}{dx^5}\right|_{0.5}\frac{(x-0.5)^5}{5!}.\end{aligned}$$

La formule précédente peut se réécrire à l'aide de valeurs moyennées sur l'élément et de leurs dérivées au centre de la cellule :

$$\begin{aligned}\mathbf{W}^h = \tilde{\mathbf{W}} + \left.\frac{d\mathbf{W}}{dx}\right|_{0.5}(x-0.5) + \left.\frac{d^2\mathbf{W}}{dx^2}\right|_{0.5}\left(\frac{(x-0.5)^2}{2} - \frac{1}{L_{st}}\int_0^1 \frac{(x-0.5)^2}{2}dx\right) \\ + \left.\frac{d^3\mathbf{W}}{dx^3}\right|_{0.5}\frac{(x-0.5)^3}{6} + \left.\frac{d^4\mathbf{W}}{dx^4}\right|_{0.5}\left(\frac{(x-0.5)^4}{24} - \frac{1}{L_{st}}\int_0^1 \frac{(x-0.5)^4}{24}dx\right) \\ + \left.\frac{d^5\mathbf{W}}{dx^5}\right|_{0.5}\frac{(x-0.5)^5}{120},\end{aligned}$$

où $\tilde{\mathbf{W}}$ est la valeur moyenne de $\mathbf{W}$ sur l'élément de référence et où L_{st} représente la taille de l'intervalle τ_{st}. On obtient ainsi facilement les six premières fonctions de base :

$$\left\{\begin{array}{lcl}\varphi_0(x) &=& 1,\\ \varphi_1(x) &=& x-0.5,\\ \varphi_2(x) &=& \dfrac{(x-0.5)^2}{2} - \dfrac{1}{24},\\ \varphi_3(x) &=& \dfrac{(x-0.5)^3}{6},\\ \varphi_4(x) &=& \dfrac{(x-0.5)^4}{24} - \dfrac{1}{1920},\\ \varphi_5(x) &=& \dfrac{(x-0.5)^5}{120}.\end{array}\right. \tag{2.3.7}$$

On visualise sur la figure 2.6 les fonctions de base de Taylor sur l'élément de référence pour les interpolations $\mathbb{P}_1$ à $\mathbb{P}_5$.

2.3.1.4 Fonctions de base de Bernstein

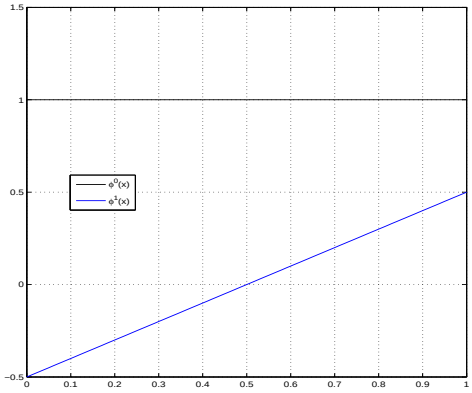
Définition 2.3.4 (Polynômes de Bernstein)
Posons $\alpha = (\alpha_1, \ldots, \alpha_{n+1}) \in I\!N^{n+1}$ tel que $\|\alpha\| = p$, les polynômes de Bernstein de degré p dans un n-simplexe s'écrivent en coordonnées barycentriques :

$$B_\alpha^p = \frac{p!}{\alpha_1! \ldots \alpha_{n+1}!}\lambda_1^{\alpha_1} \ldots \lambda_{n+1}^{\alpha_{n+1}}.$$

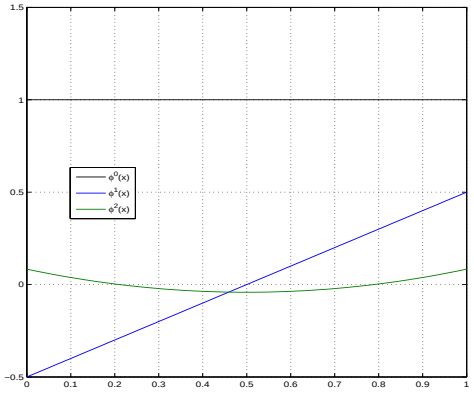
D'après ce que nous avons vu précédemment, les coordonnées barycentriques dans l'intervalle de référence $\tau_{st} = [0, 1]$ sont données par :

$$\lambda_1 = 1 - x, \quad \lambda_2 = x.$$

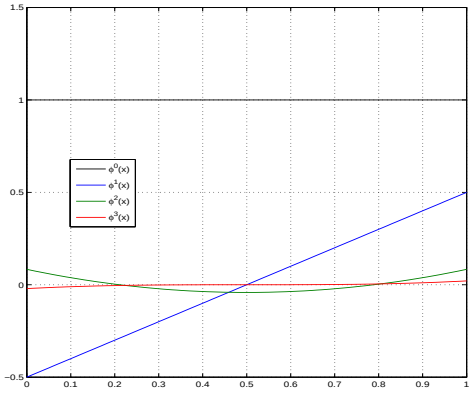
En dimension 1, on appelle base des polynômes de Bernstein d'ordre p, l'ensemble des $p + 1$ polynômes de $\mathbb{P}_p$ définis par :



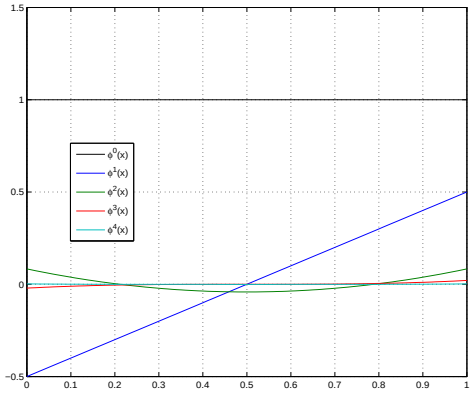
Interpolation $\mathbb{P}_1[0,1]$



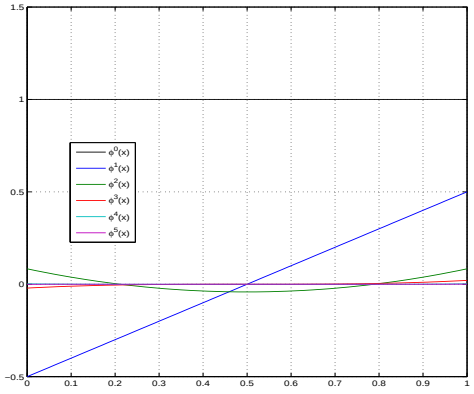
Interpolation $\mathbb{P}_2[0,1]$



Interpolation $\mathbb{P}_3[0,1]$



Interpolation $\mathbb{P}_4[0,1]$



Interpolation $\mathbb{P}_5[0,1]$

Figure 2.6 – Fonctions de base de type Taylor sur l'élément $\tau_{st} = [0,1]$

$$\varphi_k^p(x) = \left(\begin{array}{c} p \\ k \end{array} \right) \lambda_0^k \, \lambda_1^{p-k}, \quad \text{pour} \quad 0 \leq k \leq p. \tag{2.3.8}$$

Les polynômes de Bernstein forment une base modale. Ils sont issus du développement en monômes de l'équation binomiale :

$$(\lambda_i + \lambda_{i+1})^p \equiv 1.$$

Bases de Bernstein de degré 1,2,3 en dimension 1

- $\mathbf{k = 1}$
  $\varphi_1 = \lambda_1$
  $\varphi_2 = \lambda_2$
- $\mathbf{k = 2}$
  $\varphi_1 = (\lambda_1)^2$
  $\varphi_2 = 2\lambda_1\lambda_2$
  $\varphi_3 = (\lambda_2)^2$
- $\mathbf{k = 3}$
  $\varphi_1 = (\lambda_1)^3$
  $\varphi_2 = 3(\lambda_1)^2\lambda_2$
  $\varphi_3 = 3\lambda_1(\lambda_2)^2$
  $\varphi_4 = (\lambda_2)^3$

Bases de Bernstein de degré 1,2,3 en dimension 2

- $\mathbf{k = 1}$
  $\varphi_j = \lambda_j \quad j = 1, \ldots, 3$
- $\mathbf{k = 2}$
  $\varphi_j = (\lambda_j)^2 \quad j = 1, \ldots, 3$
  $\varphi_4 = 2\lambda_1\lambda_2$
  $\varphi_5 = 2\lambda_2\lambda_3$
  $\varphi_6 = 2\lambda_3\lambda_1$
- $\mathbf{k = 3}$
  $\varphi_j = (\lambda_j)^3 \quad j = 1, \ldots, 3$
  $\varphi_4 = 3(\lambda_1)^2\lambda_2$
  $\varphi_5 = 3\lambda_1(\lambda_2)^2$
  $\varphi_6 = 3(\lambda_2)^2\lambda_3$
  $\varphi_7 = 3\lambda_2(\lambda_3)^2$
  $\varphi_8 = 3(\lambda_3)^2\lambda_1$
  $\varphi_9 = 3\lambda_3(\lambda_1)^2$
  $\varphi_9 = 6\lambda_1\lambda_2\lambda_3$

Les base de polynômes de Bernstein ont la particularité d'être les bases partitionnant l'unité qui sont stables de manière optimale : une perturbation sur les coordonnées d'un polynôme implique une erreur d'appréciation du polynôme, cette erreur est mesurée par un nombre de conditionnement qui est minimal pour les bases de Bernstein.

Les polynômes de Bernstein possèdent les propriétés suivantes :

1. les polynômes de Bernstein sont homogènes *i.e.* tous leurs monônes sont de même degré ;
2. tous les polynômes de Bernstein ont la même valeur d'intégrale ;
3. $\forall p \in \mathbb{N}^*, \ \forall \mathbf{x} \in K$ on a $\sum_{j=1}^{p+1} B_j^p(\mathbf{x}) = 1$;

4. tout polynôme B_α^p défini par 2.3.4 avec $\alpha_j \neq 0$ pour $j = 1, \ldots, 3$ s'annule sur l'arête d'équation $\lambda_j^{\alpha_j} = 0$. Dans un triangle par exemple, un polynôme de Bernstein peut s'annuler sur un, deux voire les trois côtés du triangle ;

5. $\forall p \in \mathbb{N}^*,\ \forall \alpha \in \mathbb{N}^3,\ \forall \mathbf{x} \in K \quad B_\alpha^p(\mathbf{x}) \geq 0.$
Ceci est une conséquence directe de la définition des fonctions barycentriques.

On visualise sur la figure 2.7 les fonctions de base de Bernstein sur l'élément de référence $\tau_{st} = [0, 1]$ pour les interpolations $\mathbb{P}_1$ à $\mathbb{P}_5$.

2.3.2 Base nodale

Les bases nodales sont construites à partir de polynômes d'interpolation de Lagrange.

2.3.2.1 Fonctions de base de Lagrange

Définition 2.3.5 (Polynômes de Lagrange)
Soit un ensemble de $p+1$ nœuds d'interpolation notés $(x_j)_{0\leq j\leq p}$ avec $p \geq 1$, le polynôme de Lagrange $L_k^p(x)$ est l'unique polynôme d'ordre p ayant une valeur unitaire sur x_k et s'annulant sur tous les autres points nodaux x_j $(j \neq k)$. Les fonctions de Lagrange étant directement associées aux nœuds de l'élément, elles forment donc une base nodale définies comme suit :

$$L_k^p(x) = \frac{\prod_{j=0,\, j\neq k}^{p} (x - x_j)}{\prod_{j=0,\, j\neq k}^{p} (x_k - x_j)}, \quad k = 0, \cdots, p. \tag{2.3.9}$$

On observera que la famille des polynômes d'interpolation de Lagrange ne constitue pas une base hiérarchique de $\mathbb{P}_p$. En effet, si on rajoute un point d'interpolation, tous les polynômes d'interpolation de Lagrange sont modifiés.

Définition 2.3.6 (Nœuds de Lagrange)
Les points équidistants $(x_j)_{0\leq j\leq p}$ appelés nœuds de Lagrange sont définis par :

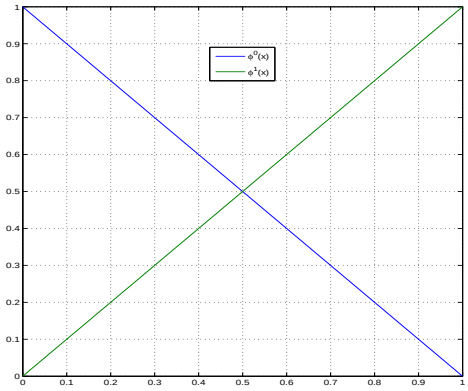
$$x_j \in \tau_{st} | \quad \lambda(x_j) \in \{0, \frac{1}{k}, \ldots, \frac{k-1}{k}, 1\}.$$

Remarque : Si l'on note $g(x)$ le polynôme d'ordre $p+1$ s'annulant sur les $p+1$ points nodaux $(x_j)_{0\leq j\leq p}$, alors on peut écrire $L_k^p(x)$ dans une forme plus compacte :

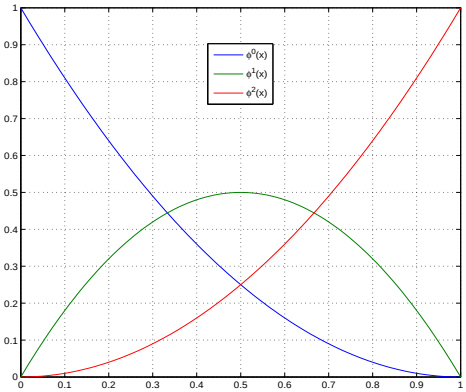
$$L_k^p(x) = \frac{g(x)}{g'(x)(x - x_k)}.$$

Les polynômes de Lagrange partagent avec les polynômes de Bernstein les propriétés suivantes :

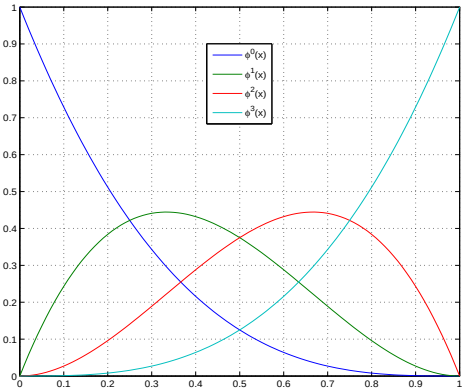
1. les polynômes de Lagrange sont homogènes *i.e.* tous leurs monônes sont de même degré ;
2. $\forall p \in \mathbb{N}^*,\ \forall \mathbf{x} \in K$ on a $\sum_{j=1}^{n+1} L_j^K(\mathbf{x}) = 1$;
3. tout polynôme L_j^p s'annule sur le ou les côtés du triangle qui ne contiennent pas le nœud x_j correspondant. La conséquence d'une telle propriété est de faciliter les calculs à l'interface entre deux triangles (*i.e.* une arête) car seuls les polynômes non identiquement nuls sur cette arête seront considérés dans ces calculs.



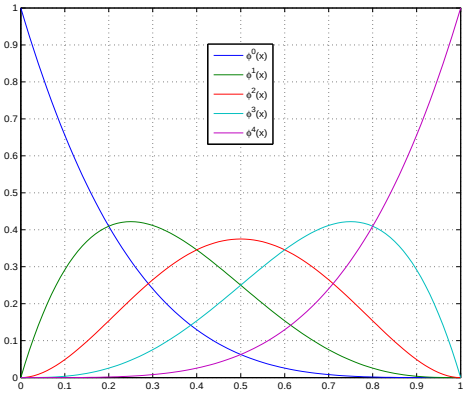
Interpolation $\mathbb{P}_1[0,1]$



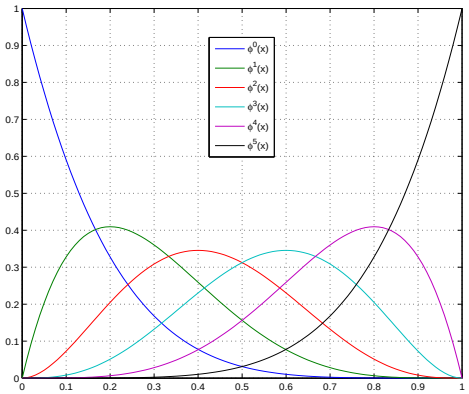
Interpolation $\mathbb{P}_2[0,1]$



Interpolation $\mathbb{P}_3[0,1]$



Interpolation $\mathbb{P}_4[0,1]$



Interpolation $\mathbb{P}_5[0,1]$

Figure 2.7 – Fonctions de base de Bernstein sur l'élément $\tau_{st} = [0,1]$

D'après la définition (2.3.5), les polynômes de Lagrange possèdent la propriété de collocation, *i.e.* $L_m^p(x_n) = \delta_{mn}, \quad m,n \in \{0,\dots,p\}$ où δ_{mn} représente le symbole de Kronecker, et constituent de ce fait une base d'interpolation particulièrement utile par exemple lors de l'inversion d'une *matrice de Vandermonde*, introduite lorsque nous avons étudié les fonctions de base canoniques, établissant la connection entre les modes $\widetilde{\mathbf{W}_k}$, et les valeurs nodales $(\mathbf{W}(x_j))_{0\le j\le p}$.

Définition 2.3.7 (Opérateur d'interpolation local de Lagrange)
L'opérateur d'interpolation local de Lagrange $\mathcal{I}_K^{Lag}$ est défini comme suit :

$$\mathcal{I}_K^{Lag} : \mathcal{C}^0(K) \ni \mathbf{W} \to \sum_{k=1}^{p} \widetilde{\mathbf{W}_k}\, L_k^p(x) \in \mathbb{P}_p.$$

On dit que $\mathcal{I}_K^{Lag}\mathbf{W}$ est *l'interpolé de Lagrange* de $\mathbf{W}$ sur K.

Puisque l'approximation polynomiale passe par tous les points du maillage, on sait que $\mathcal{I}\mathbf{W}(x_j) = \mathbf{W}(x_j) \quad \forall j = 0,\cdots,p$ et, grâce à la propriété de collocation, cela signifie que l'interpolé de Lagrange est tel que sa valeur aux nœuds $(x_j)_{0\le j\le p}$ coïncide avec celle de la fonction à interpoler $\mathbf{W}$ *i.e.* $\widetilde{\mathbf{W}_k} = \mathbf{W}(x_k)$. Puisque les coordonnées de $\mathbf{W}_i$ dans la base des polynômes de Lagrange sont simplement donnés par les valeurs ponctuelles $(y_k)_{0\le k\le p}$ de la fonction à interpoler, la matrice de Vandermonde introduite dans le cadre des fonctions de base canonique se simplifie en une matrice diagonale facile à inverser.

L'approximation polynomiale devient alors :

$$\mathcal{I}_K^{Lag}\mathbf{W}(x) = \sum_{k=1}^{p} \mathbf{W}(x_k)\, L_k^p(x).$$

L'opérateur d'interpolation $\mathcal{I}_K^{Lag}$ est une *projection* de $\mathcal{C}^0(K)$ dans $\mathbb{P}_p$. En effet, pour tout $p \in \mathbb{P}_p$, en décomposant p dans la base des fonctions de base selon $p = \sum_{l=1}^{p} p_l L_l^p$, on obtient :

$$\mathcal{I}_K^{Lag}\, p = \sum_{k=1}^{p} p(x_k) L_k^p = \sum_{k,l=1}^{p} p_l L_l^p(x_k) L_k^p = \sum_{k,l=1}^{p} p_l \delta_{kl} L_k^p = p.$$

Par suite, pour tout $v \in \mathcal{C}^0(K)$, il vient $\mathcal{I}_K^{Lag}(\mathcal{I}_K^{Lag}\, v) = \mathcal{I}_K^{Lag}\, v$.
Cette transformation d'un polynôme à une interpolation Lagrangienne servira particulièrement dans le calcul d'intégrales ou de dérivées faisant intervenir des représentations modales.

On munit $\mathbb{P}_p$ de la norme canonique $\|.\|_{\mathcal{C}^0(K)}$ définie pour $v \in \mathcal{C}^0(K)$ par $\|v\|_{\mathcal{C}^0(K)} = \sup_{x\in K} |v(x)|$. Un calcul simple montre que pour l'opérateur d'interpolation de Lagrange $\mathcal{I}_K^{Lag}$, on a :

$$\|\mathcal{I}_K^{Lag}\|_{\mathcal{L}(\mathcal{C}^0(K),\mathbb{P}_p)} = \sup_{v\in\mathcal{C}^0(K)} \frac{\|\mathcal{I}_K^{Lag}\, v\|_{\mathcal{C}^0(K)}}{\|v\|_{\mathcal{C}^0(K)}} = \sup_x \left(\sum_{k=1}^{p} |L_k^p(x)| \right). \tag{2.3.10}$$

Cette quantité ne dépend que des points d'interpolation $(x_j)_{0\le j\le p}$ et est appelée *la constante de Lebesgue* associée à ces nœuds, notée Λ. On s'aperçoit que :

$$\begin{aligned} \|\mathbf{W} - \mathbf{W}^h\|_\infty &= \|\mathbf{W} - \mathbf{W}^* + \mathbf{W}^* - \mathbf{W}^h\|_\infty, \\ &\le \|\mathbf{W} - \mathbf{W}^*\|_\infty + \|\mathbf{W}^* - \mathbf{W}^h\|_\infty, \\ &\le (1+\Lambda)\|\mathbf{W} - \mathbf{W}^*\|_\infty, \end{aligned}$$

où $\|.\|_\infty$ est la norme du maximum usuelle et $\mathbf{W}^*$ représente la meilleure approximation polynomiale d'ordre p. Par conséquent, la constante de Lebesgue est donc liée au problème de la convergence des polynômes d'interpolation et peut s'interpréter comme le facteur d'amplification de l'erreur dans le

procédé d'interpolation de Lagrange. On remarque que la valeur de Λ se détermine à partir des points d'interpolation $(x_j)_{0\leq j\leq p}$, c'est à dire que la meilleure représentation polynomiale sera obtenue pour un jeu de points de collocation qui minimisera la constante de Lebesgue. L'interpolation de Lagrange en des points équidistants, par exemple, n'est pas une méthode numérique très stable car elle dépend beaucoup de la fonction que l'on cherche à représenter. Le nombre de conditionnement de la matrice de masse, *i.e.* le rapport entre la plus grande et la plus petite valeur propre de la matrice, explose exponentiellement en p. De plus, la constante de Lebesgue associée aux nœuds équirépartis explose, elle aussi, exponentiellement en p. Généralement, dès que le degré polynomial devient grand ou qu'on augmente le nombre de points, on constate que le polynôme se met à osciller fortement entre les noeuds du maillage avec une amplitude de plus en plus grande, c'est ce qu'on appelle le *phénomène de Runge*. Comme nous l'avons mentionné précédemment, des répartitions autres que celle uniforme permettent toutefois d'améliorer substantiellement la précision de l'interpolé lorsqu'on utilise des polynômes de degré élevé.

Définition 2.3.8 (Points de Gauss-Lobatto)
Soit un entier $p \geq 1$. Les $(p+1)$ points de Gauss-Lobatto $\{g_0^p, \ldots, g_p^p\}$ sur l'intervalle de référence $K = [0,1]$ sont les deux extrémités de l'intervalle, $g_0^p = 0$ et $g_p^p = 1$, et les $(p-1)$ racines de la dérivée du polynôme de Legendre $G_p^{'}$.

On notera que comme G_p admet p racines distinctes sur K, $G_p^{'}$ admet $(p-1)$ racines distinctes sur K. Les points de Gauss-Lobatto sur K sont portés dans le tableau 2.3.1 pour $p \in \{1,2,3,4\}$.

	p = 1	**p = 2**	**p = 3**	**p = 4**
l = 0	0	0	0	0
l = 1	1	$\frac{1}{2}$	$\frac{1}{2}(1-(\frac{1}{5})^{\frac{1}{2}})$	$\frac{1}{2}(1-(\frac{3}{7})^{\frac{1}{2}})$
l = 2		1	$\frac{1}{2}(1+(\frac{1}{5})^{\frac{1}{2}})$	$\frac{1}{2}$
l = 3			1	$\frac{1}{2}(1+(\frac{3}{7})^{\frac{1}{2}})$
l = 4				1

TABLE 2.3.1 – Points de Gauss-Lobatto sur l'intervalle de référence $\tau_{st} = [0,1]$ pour $p \in \{1,2,3,4\}$

Ces points de collocation sont irrégulièrement répartis sur l'intervalle, ils sont en particulier plus resserrés vers les frontières de l'intervalle. Les bornes de l'intervalle, ici 0 et 1, font partie des points de collocation mais il existe d'autres jeux de points (dits de Gauss et Gauss-Radau) qui incluent ou non l'une ou/et l'autre des extrémités du domaine.

Les $(p+1)$ polynômes d'interpolation de Lagrange associés aux $(p+1)$ points de Gauss-Lobatto $\{g_0^p, \ldots, g_p^p\}$ sur l'intervalle de référence $K = [0,1]$ sont notés $\{\theta_0^{GL,p}, \ldots, \theta_p^{GL,p}\}$. Ces polynômes sont tels que :

$$\theta_m^{GL,p}(x) = \frac{(x-1)\,x\,G_p^{'}(x)}{p(p+1)G_p(g_m^p)(x-g_m^p)}, \quad m \in \{0,\ldots,p\}.$$

La famille $\{\theta_0^{GL,p}, \ldots, \theta_p^{GL,p}\}$ forme une base nodale de $\mathbb{P}_p$. Pour $p \in \{1,2\}$, la base nodale est constituée des polynômes d'interpolation de Lagrange usuels puisque les points de Gauss-Lobatto sont équirépartis sur $[0,1]$. La base nodale $\{\theta_0^{GL,p}, \ldots, \theta_p^{GL,p}\}$ n'est pas hiérarchique mais présente l'avantage d'approcher les coefficients de la matrice de masse en utilisant la quadrature de Gauss-Lobatto pour transformer une matrice dense en une matrice diagonale. Enfin, un résultat dû à Fejér (1932) est que la base nodale $\{\theta_0^{GL,p}, \ldots, \theta_p^{GL,p}\}$ est telle que :

$$\sup_x \left(\sum_{m=0}^{p} (\theta_m^{GL,p}(x))^2 \right) = 1.$$

Cette propriété remarquable implique en particulier que :

$$\sup_x |\theta_m^{GL,p}(x)| = 1, \quad m \in \{0, \ldots, p\}.$$

En d'autre termes, le polynôme $\theta_m^{GL,p}$ atteint son maximum sur K au point de Gauss-Lobatto g_m^p. On observera que cette propriété n'est pas satisfaite par les polynômes d'interpolation de Lagrange puisque ceux-ci peuvent prendre des valeurs plus grandes que 1 sur K d'après (2.8). En effet, on déduit immédiatement de la formule (2.3.10) que :

$$\Lambda(\{g_0^p, \ldots, g_p^p\}) = \sup_x \left(\sum_{m=0}^{p} |\theta_m^{GL,p}(x)| \right) \leq (p+1).$$

Même si cette estimation est quelque peu pessimiste, elle représente déjà une amélioration considérable par rapport à la constante de Lebesgue associée aux nœuds équirépartis sur K, cette dernière explosant exponentiellement en k.

Lorsque K est un hypercube en dimension $d \geq 2$, on peut construire une base sur K à partir de produits tensoriels des éléments de la base unidimensionnelle. La construction de bases sur des simplexes est plus technique car elle utilise des transformations non-linéaires ; voir Karniadakis et Spencer [Karniadakis 2005].

Définition 2.3.9 (Elément fini spectral)
Un élément fini dont la base nodale est constituée des polynômes d'interpolation de Lagrange associés aux points de Gauss-Lobatto en dimension 1 ou à des produits tensoriels de ceux-ci en dimension $d \geq 2$ (lorsque K est un hypercube) est appelé un élément fini spectral.

En pratique, il est plus judicieux d'exprimer les polynômes de Lagrange comme des polynômes de degré $\leq p$ des fonctions barycentriques λ_i, $i = 1, 2, 3$, considérées comme les variables de ces polynômes. On notera que pour $p = 1$, les $(p+1)$ fonctions de base de Lagrange sont les coordonnées barycentriques. On donne ci-dessous l'expression de ces polynômes pour les degrés un, deux et trois dans un simplexe de dimension 1 puis 2.

Bases de Lagrange de degré 1,2,3 en dimension 1

- $\mathbf{k = 1}$
  $\varphi_1 = \lambda_1$
  $\varphi_2 = \lambda_2$
- $\mathbf{k = 2}$
  $\varphi_1 = \lambda_1(2\lambda_1 - 1)$
  $\varphi_2 = 4\lambda_1\lambda_2$
  $\varphi_3 = \lambda_2(2\lambda_2 - 1)$
- $\mathbf{k = 3}$
  $\varphi_1 = \lambda_1(3\lambda_1 - 1)(3\lambda_1 - 2)/2$
  $\varphi_2 = 9\lambda_1\lambda_2(3\lambda_1 - 1)/2$
  $\varphi_3 = 9\lambda_1\lambda_2(3\lambda_2 - 1)/2$
  $\varphi_3 = \lambda_2(3\lambda_2 - 1)(3\lambda_2 - 2)/2$

Bases de Lagrange de degré 1,2,3 en dimension 2

- $\mathbf{k = 1}$
  $\varphi_j = \lambda_j \quad j = 1, \ldots, 3$
- $\mathbf{k = 2}$
  $\varphi_1 = \lambda_1(2\lambda_1 - 1)$
  $\varphi_2 = \lambda_2(2\lambda_2 - 1)$
  $\varphi_3 = \lambda_3(2\lambda_3 - 1)$
  $\varphi_4 = 4\lambda_1\lambda_2$

$\varphi_5 = 4\lambda_2\lambda_3$
$\varphi_6 = 4\lambda_1\lambda_3$

– $\mathbf{k = 3}$
$\varphi_1 = \lambda_1(3\lambda_1 - 1)(3\lambda_1 - 2)/2$
$\varphi_2 = \lambda_2(3\lambda_2 - 1)(3\lambda_2 - 2)/2$
$\varphi_3 = \lambda_3(3\lambda_3 - 1)(3\lambda_3 - 2)/2$
$\varphi_4 = 9\lambda_1\lambda_2(3\lambda_1 - 1)/2$
$\varphi_5 = 9\lambda_1\lambda_2(3\lambda_2 - 1)/2$
$\varphi_6 = 9\lambda_2\lambda_3(3\lambda_2 - 1)/2$
$\varphi_7 = 9\lambda_2\lambda_3(3\lambda_3 - 1)/2$
$\varphi_8 = 9\lambda_1\lambda_3(3\lambda_3 - 1)/2$
$\varphi_9 = 9\lambda_1\lambda_3(3\lambda_1 - 1)/2$
$\varphi_{10} = 27\lambda_1\lambda_2\lambda_3$

On visualise sur la figure 2.8 les fonctions de base de Lagrange sur l'élément de référence $\tau_{st} = [0,1]$ pour les interpolations $\mathbb{P}_1$ à $\mathbb{P}_5$.

2.4 Etude numérique en 1D

On se propose ici de réaliser une étude numérique comparative détaillée des différentes bases d'interpolation polynomiale introduites dans la section 2.3 dans le contexte de la discrétisation des équations de Maxwell 1D sur l'intervalle $\tau_{st} = [0,1]$ par une méthode Galerkin discontinue. Dans ce qui suit, pour simplifier la présentation, on suppose que l'ordre d'approximation est identique pour tous les éléments du maillage *i.e.* $p_i = p$ pour $1 \leq i \leq N_{\mathcal{T}_h}$.

2.4.1 Equations de Maxwell 1D

Afin d'obtenir une version 1D des équations de Maxwell (1.1.20) nous considérons comme direction de propagation $\mathbf{k} = (k,0,0)^t$ avec une polarisation du vecteur champ électrique telle que $\mathbf{E} = (0,0,E_z)^t$. La polarisation du vecteur champ magnétique se déduit du produit vectoriel $\mathbf{k} \times \mathbf{E}$ menant à $\mathbf{H} = (0,H_y,0)^t$. De plus, nous supposons que E_z et H_y sont fonctions de x et t. Pour plus de clarté, nous allégeons les notations en remplaçant E_z par E et H_y par H. Les équations de Maxwell 1D sont alors données par :

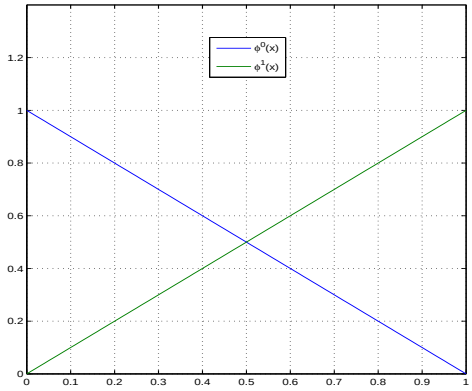
$$\left\{ \begin{array}{rcl} \varepsilon \dfrac{\partial E}{\partial t} - \dfrac{\partial H}{\partial x} & = & -z_0\sigma E, \\ \mu \dfrac{\partial H}{\partial t} - \dfrac{\partial E}{\partial x} & = & 0. \end{array} \right. \tag{2.4.1}$$

Ce système peut être réécrit sous la forme :

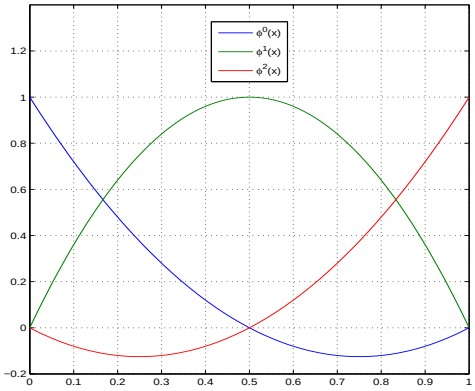
$$Q\mathbf{W}_t + \partial_x F(\mathbf{W}) + K\mathbf{W} = 0, \tag{2.4.2}$$

où $\mathbf{W} = \begin{pmatrix} E \\ H \end{pmatrix}$, $\mathbf{W}_t = \dfrac{\partial \mathbf{W}}{\partial t}$, $\partial_x F(\mathbf{W}) = \dfrac{\partial F(\mathbf{W})}{\partial x}$ et :

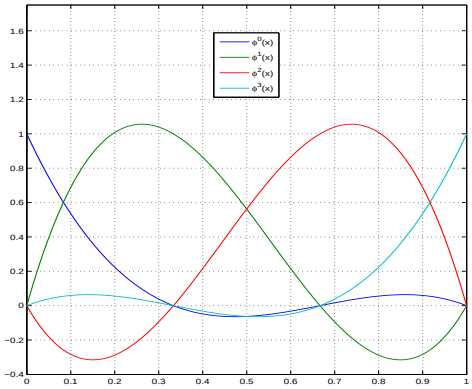
$$Q = \begin{bmatrix} \varepsilon & 0 \\ 0 & \mu \end{bmatrix}, \; K = \begin{pmatrix} z_0\sigma \\ 0 \end{pmatrix} \quad \text{et} \quad F(\mathbf{W}) = \begin{pmatrix} -H \\ -E \end{pmatrix} = A\mathbf{W} \quad \text{avec} \quad A = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}.$$



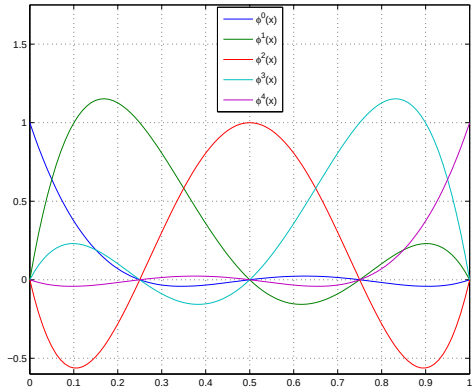
Interpolation $\mathbb{P}_1[0,1]$



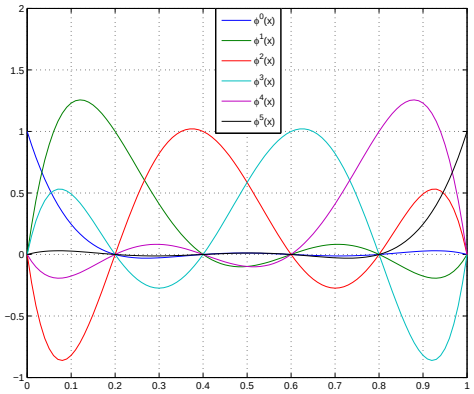
Interpolation $\mathbb{P}_2[0,1]$



Interpolation $\mathbb{P}_3[0,1]$



Interpolation $\mathbb{P}_4[0,1]$



Interpolation $\mathbb{P}_5[0,1]$

Figure 2.8 – Fonctions de base de Lagrange sur l'élément $\tau_{st} = [0,1]$

2.4.2 Discrétisation spatiale

2.4.2.1 Formulation faible

En 1D, le domaine Ω est un intervalle $\Omega = [x_0, x_0 + L]$ discrétisé par un ensemble de N segments τ_i. On a $\Omega = \bigcup_{i=1}^{N} \tau_i$, où $\tau_i = [x_{i-1}, x_i]$ avec $x_N = x_0 + L$ (voir la figure 2.9).

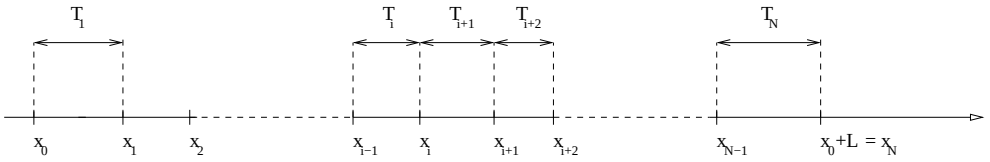


Figure 2.9 – Maillage en 1D.

On note φ une fonction test scalaire. Nous partons donc de l'équation (2.4.2) et nous supposons que la densité de courant est nulle et que les caractéristiques électromagnétiques ε_i et μ_i sont constantes sur τ_i :

$$Q\mathbf{W}_t + \partial_x F(\mathbf{W}) = 0,$$

En multipliant cette équation par φ, en intégrant sur τ_i et en intégrant par parties, nous obtenons :

$$\begin{aligned} &\int_{\tau_i} Q\mathbf{W}_t\,\varphi\,dx + \int_{\tau_i} (\partial_x F(\mathbf{W}))\,\varphi\,dx = 0, \\ \Leftrightarrow\quad &\int_{\tau_i} Q\mathbf{W}_t\,\varphi\,dx - \int_{\tau_i} F(\mathbf{W})\partial_x\varphi\,dx + [F(\mathbf{W})\varphi]_{x_{i-1}}^{x_i} = 0. \end{aligned} \tag{2.4.3}$$

Soit $\mathcal{P}_i = \mathbb{P}_p[\tau_i]$ l'espace des fonctions polynomiales de degré au plus p sur τ_i. Les degrés de liberté locaux sont notés $\mathbf{W}_{ij} \in \mathbb{R}^2$. Soit $\phi_i = (\varphi_{i1}, \varphi_{i2}, \cdots, \varphi_{id_i})$ une base locale de $\mathcal{P}_i$ et $\mathbf{W}_i$ la projection L_2-orthogonale de $\mathbf{W}$ sur $\mathcal{P}_i$ définie par (1.2.2). En injectant $\mathbf{W}_i$ dans les intégrales volumiques de (2.4.3) et en tenant compte de la linéarité de $F(\mathbf{W})$ nous obtenons :

$$\begin{aligned} (2.4.3)\quad \Rightarrow\quad &\int_{\tau_i} Q(\mathbf{W}_i)_t\,\varphi\,dx - \int_{\tau_i} F(\mathbf{W}_i)\partial_x\varphi\,dx + [F(\mathbf{W})\varphi]_{x_{i-1}}^{x_i} = 0 \\ \Leftrightarrow\quad &Q_i\int_{\tau_i} (\mathbf{W}_i)_t\varphi\,dx - \int_{\tau_i} F(\mathbf{W}_i)\partial_x\varphi\,dx + \varphi(x_i)F(\mathbf{W})_{|x_i} - \varphi(x_{i-1})F(\mathbf{W})_{|x_{i-1}} = 0, \end{aligned} \tag{2.4.4}$$

où :

$$Q_i = \begin{bmatrix} \varepsilon_i & 0 \\ 0 & \mu_i \end{bmatrix}.$$

Dans (2.4.4), $x_i = \tau_i \cap \tau_{i+1}$ est le point commun entre les segments (successifs) τ_i et τ_{i+1}. Similairement au cas 3D, puisque la représentation locale de $\mathbf{W}$ est discontinue d'un élément à un autre, un traitement particulier doit être introduit pour l'évaluation de $F(\mathbf{W})$ au point x_i et on utilise un flux numérique basé sur un schéma centré :

$$F(\mathbf{W})_{|x_i} \approx \frac{F(\mathbf{W})_i^i + F(\mathbf{W})_{i+1}^i}{2}, \quad \text{où} \quad F(\mathbf{W})_i^k = F(\mathbf{W}_i(x_k)) = A\mathbf{W}_i(x_k),$$

ce qui conduit à :

$$
\begin{aligned}
(2.4.4) \Rightarrow Q_i \int_{\tau_i} (\mathbf{W}_i)_t \, \varphi \, dx \quad &- \quad \int_{\tau_i} F(\mathbf{W}_i) \partial_x \varphi \, dx \\
&+ \quad \frac{1}{2}(A\mathbf{W}_i(x_i) + A\mathbf{W}_{i+1}(x_i))\varphi(x_i) \\
&- \quad \frac{1}{2}(A\mathbf{W}_{i-1}(x_{i-1}) + A\mathbf{W}_i(x_{i-1}))\varphi(x_{i-1}) = 0,
\end{aligned} \tag{2.4.5}
$$

c'est à dire :

$$
\begin{aligned}
Q_i \int_{\tau_i} (\mathbf{W}_i)_t \, \varphi \, dx - \int_{\tau_i} F(\mathbf{W}_i) \partial_x \varphi \, dx \quad &- \quad \frac{1}{2} A\mathbf{W}_{i-1}(x_{i-1})\varphi(x_{i-1}) + \frac{1}{2} A\mathbf{W}_{i+1}(x_i)\varphi(x_i) \\
&- \quad \frac{1}{2}[A\mathbf{W}_i(x)\varphi(x)]_{x_{i-1}}^{x_i} = 0.
\end{aligned}
$$

En utilisant la formule d'intégration par parties :

$$
[A\mathbf{W}_i(x)\varphi(x)]_{x_{i-1}}^{x_i} = \int_{\tau_i} A\mathbf{W}_i(x)\partial_x\varphi(x)dx + \int_{\tau_i} A\partial_x\mathbf{W}_i(x)\varphi(x)dx,
$$

on obtient, $\forall \varphi \in \tau_i$:

$$
\begin{aligned}
Q_i \int_{\tau_i} \frac{\partial \mathbf{W}_i}{\partial t} \, \varphi \, dx \quad &+ \quad \frac{1}{2} \int_{\tau_i} (A \frac{\partial \mathbf{W}_i}{\partial x} \, \varphi - A\mathbf{W}_i \frac{\partial \varphi}{\partial x}) dx \\
&- \quad \frac{1}{2} A\mathbf{W}_{i-1}(x_{i-1})\varphi(x_{i-1}) + \frac{1}{2} A\mathbf{W}_{i+1}(x_i)\varphi(x_i) = 0,
\end{aligned} \tag{2.4.6}
$$

où on a de plus supposé que les caractéristiques électromagnétiques ε_i et μ_i sont constantes sur τ_i.

2.4.2.2 Traitement numérique des conditions aux limites

En 1D, les frontières du domaine de calcul sont en x_0 et x_N. En toute généralité, on peut distinguer quatre cas :

- **cas 1 :** si $x_0 \in \mathcal{F}_m$ et $x_N \in \mathcal{F}_m$ alors nous posons :

$$
\mathbf{W}_0 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_1 \quad \text{et} \quad \mathbf{W}_{N+1} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_N.
$$

  Alors, $\mathbf{W}_{|x_0} = \dfrac{\mathbf{W}_0^0 + \mathbf{W}_1^0}{2}$ et $\mathbf{W}_{|x_N} = \dfrac{\mathbf{W}_N^N + \mathbf{W}_{N+1}^N}{2}$ sont tels que :

$$
\mathbf{W}_{|x_0} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_1^0 \quad \text{et} \quad \mathbf{W}_{|x_N} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_N^N.
$$

- **cas 2 :** si $x_0 \in \mathcal{F}_m$ et $x_N \in \mathcal{F}_a$ alors nous posons :

$$
\mathbf{W}_0 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_1 \quad \text{et} \quad \mathbf{W}_{N+1} = \begin{bmatrix} 0 & -z_N \\ -z_N^{-1} & 0 \end{bmatrix} \mathbf{W}_N + \begin{bmatrix} 1 & -z_N \\ z_N^{-1} & 1 \end{bmatrix} \mathbf{W}_N^\infty,
$$

  où $\mathbf{W}_N^\infty$ est la projection de $\mathbf{W}^\infty$ sur τ_N.

  Alors, $\mathbf{W}_{|x_0} = \dfrac{\mathbf{W}_0^0 + \mathbf{W}_1^0}{2}$ et $\mathbf{W}_{|x_N} = \dfrac{\mathbf{W}_N^N + \mathbf{W}_{N+1}^N}{2}$ sont tels que :

$$
\mathbf{W}_{|x_0} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_1^0 \quad \text{et} \quad \mathbf{W}_{|x_N} = \frac{1}{2} \begin{bmatrix} 1 & -z_N \\ -z_N^{-1} & 1 \end{bmatrix} \mathbf{W}_N^N + \frac{1}{2} \begin{bmatrix} 1 & -z_N \\ z_N^{-1} & 1 \end{bmatrix} \mathbf{W}_N^\infty.
$$

– **cas 3 :** si $x_0 \in \mathcal{F}_a$ et $x_N \in \mathcal{F}_m$ alors nous posons :

$$\mathbf{W}_0 = \begin{bmatrix} 0 & z_1 \\ z_1^{-1} & 0 \end{bmatrix} \mathbf{W}_1 + \begin{bmatrix} 1 & z_1 \\ -z_1^{-1} & 1 \end{bmatrix} \mathbf{W}_1^{\infty} \quad \text{et} \quad \mathbf{W}_{N+1} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_N,$$

où $\mathbf{W}_1^{\infty}$ est la projection de $\mathbf{W}^{\infty}$ sur τ_1.

Alors, $\mathbf{W}_{|x_0} = \dfrac{\mathbf{W}_0^0 + \mathbf{W}_1^0}{2}$ et $\mathbf{W}_{|x_N} = \dfrac{\mathbf{W}_N^N + \mathbf{W}_{N+1}^N}{2}$ sont tels que :

$$\mathbf{W}_{|x_0} = \frac{1}{2}\begin{bmatrix} 1 & z_1 \\ z_1^{-1} & 1 \end{bmatrix} \mathbf{W}_1^0 + \frac{1}{2}\begin{bmatrix} 1 & z_1 \\ -z_1^{-1} & 1 \end{bmatrix} \mathbf{W}_1^{\infty} \quad \text{et} \quad \mathbf{W}_{|x_N} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{W}_N^N.$$

– **cas 4 :** si $x_0 \in \mathcal{F}_a$ et $x_N \in \mathcal{F}_a$ alors nous posons :

$$\mathbf{W}_0 = \begin{bmatrix} 0 & z_1 \\ z_1^{-1} & 0 \end{bmatrix} \mathbf{W}_1 + \begin{bmatrix} 1 & z_1 \\ -z_1^{-1} & 1 \end{bmatrix} \mathbf{W}_1^{\infty},$$

et

$$\mathbf{W}_{N+1} = \begin{bmatrix} 0 & -z_N \\ -z_N^{-1} & 0 \end{bmatrix} \mathbf{W}_N + \begin{bmatrix} 1 & -z_N \\ z_N^{-1} & 1 \end{bmatrix} \mathbf{W}_N^{\infty}.$$

Alors, $\mathbf{W}_{|x_0} = \dfrac{\mathbf{W}_0^0 + \mathbf{W}_1^0}{2}$ et $\mathbf{W}_{|x_N} = \dfrac{\mathbf{W}_N^N + \mathbf{W}_{N+1}^N}{2}$ sont tels que :

$$\mathbf{W}_{|x_0} = \frac{1}{2}\begin{bmatrix} 1 & z_1 \\ z_1^{-1} & 1 \end{bmatrix} \mathbf{W}_1^0 + \frac{1}{2}\begin{bmatrix} 1 & z_1 \\ -z_1^{-1} & 1 \end{bmatrix} \mathbf{W}_1^{\infty},$$

et

$$\mathbf{W}_{|x_N} = \frac{1}{2}\begin{bmatrix} 1 & -z_N \\ -z_N^{-1} & 1 \end{bmatrix} \mathbf{W}_N^N + \frac{1}{2}\begin{bmatrix} 1 & -z_N \\ z_N^{-1} & 1 \end{bmatrix} \mathbf{W}_N^{\infty}.$$

2.4.2.3 Equations semi-discrétisées

En remplaçant φ par φ_{il} dans (2.4.5) pour $1 \leq l \leq d_i$, $1 \leq i \leq N$, nous obtenons :

$$\begin{aligned} Q_i \int_{\tau_i} \frac{\partial \mathbf{W}_i}{\partial t}\varphi_{il} dx - \int_{\tau_i} A\mathbf{W}_i \frac{\partial \varphi_{il}}{\partial x} dx \quad &- \quad \frac{1}{2}(A\mathbf{W}_{i-1}(x_{i-1}) + A\mathbf{W}_i(x_{i-1}))\varphi_{il}(x_{i-1}) \\ &+ \quad \frac{1}{2}(A\mathbf{W}_i(x_i) + A\mathbf{W}_{i+1}(x_i))\varphi_{il}(x_i) = 0, \end{aligned}$$

soit en développant à l'aide de la formule (1.2.2) :

$$\begin{aligned} & Q_i \int_{\tau_i} \sum_{j=1}^{d_i} \left(\mathbf{W}_{ij} \frac{\partial \varphi_{ij}}{\partial t}\right) \varphi_{il} dx - \int_{\tau_i} \sum_{j=1}^{d_i} (A\mathbf{W}_{ij}\varphi_{ij}) \frac{\partial \varphi_{il}}{\partial x} dx \\ - \quad & \frac{1}{2}\left[\sum_{j=1}^{d_{i-1}} (A\mathbf{W}_{i-1,j}\varphi_{i-1,j})(x_{i-1}) + \sum_{j=1}^{d_i} (A\mathbf{W}_{ij}\varphi_{ij})(x_{i-1})\right] \varphi_{il}(x_{i-1}) \\ + \quad & \frac{1}{2}\left[\sum_{j=1}^{d_i} (A\mathbf{W}_{ij}\varphi_{ij})(x_i) + \sum_{j=1}^{d_{i+1}} (A\mathbf{W}_{i+1,j}\varphi_{i+1,j})(x_i)\right] \varphi_{il}(x_i) = 0, \end{aligned} \tag{2.4.7}$$

soit encore sous forme matricielle locale :

$$Q_i(\mathbf{W}_i)_t\Phi_i - A\mathbf{W}_i\Phi_i^* + \frac{A\mathbf{W}_i\Phi_i^i + A\mathbf{W}_{i+1}\Phi_{i+1}^i}{2} - \frac{A\mathbf{W}_i\Phi_i^{i-1} + A\mathbf{W}_{i-1}\Phi_{i-1}^{i-1}}{2} = 0, \tag{2.4.8}$$

avec :

$$\begin{cases} \Phi_i & = & \displaystyle\int_{\tau_i} {}^t\phi_i\phi_i dx, \\ \Phi_i^* & = & \displaystyle\int_{\tau_i} {}^t\left(\partial_{x_k}\phi_i\right)\phi_i dx, \\ \Phi_i^i & = & {}^t\phi_i(x_i)\phi_i(x_i), \\ \Phi_{i+1}^i & = & {}^t\phi_i(x_i)\phi_{i+1}(x_i), \\ \Phi_i^{i-1} & = & {}^t\phi_i(x_{i-1})\phi_i(x_{i-1}), \\ \Phi_{i-1}^{i-1} & = & {}^t\phi_i(x_{i-1})\phi_{i-1}(x_{i-1}), \end{cases}$$

où :

- $\phi_i = (\varphi_{i1}, \varphi_{i2}, \cdots, \varphi_{id_i})$ (ϕ_i est un vecteur $d_i \times 1$),
- Φ_i est une matrice $d_i \times d_i$ symétrique définie positive,
- Φ_i^* est une matrice $d_i \times d_i$.

2.4.3 Intégration en temps

Deux types de schéma sont généralement utilisés pour résoudre des problèmes évolutifs :

- les schémas explicites. Le principal avantage de ces schémas est la facilité de mise en œuvre. En général, il est aussi relativement aisé de construire des schémas explicites d'ordre élevé. En revanche, la stabilité de ces schémas est contrainte par une condition de stabilité qui peut s'avérer très restrictive lorsque le maillage est non-uniforme (localement raffiné) ou/et si l'approximation repose sur une interpolation d'ordre élevé. Parmi ces schémas, les plus utilisés pour la résolution numérique des équations de Maxwell sont les schémas de type Runge-Kutta et les schémas de type saute-mouton.
- les schémas implicites. Le plus souvent, ces schémas sont inconditionnellement stables. Cependant, ils conduisent à la résolution d'un système linéaire à chaque pas de temps ce qui implique des surcoûts en termes de temps de calcul et d'occupation mémoire. Le choix ou la mise au point de la méthode de résolution de ce système linéaire est une étape importante qui conditionne notablement la pertinence d'un schéma implicite.

Dans cette étude, on se limite à l'utilisation de schémas explicites.

2.4.3.1 Schémas de type saute-mouton

Un schéma d'intégration en temps très utilisé pour les équations de Maxwell est le schéma *saute-mouton*. Celui-ci est notamment adopté dans la méthode différence finie en domaine temporel due à Yee [Yee 1966]. Dans sa forme la plus courante, il s'agit d'un schéma explicite centré et précis au second ordre en temps. En fait, il est possible de formuler une famille de schémas saute-mouton d'ordre arbitrairement élevé. Nous considérons ici la famille de schémas étudiée dans [Spachmann 2002] et plus particulièrement le schéma saute-mouton du quatrième ordre.

Le schéma saute-mouton du second ordre. Les champs $\mathbf{H}_h$ et $\mathbf{E}_h$ sont évalués sur une grille décalée en temps. Le schéma s'écrit :

$$\left\{\begin{array}{lcll} \mathbf{H}_h^{n+\frac{3}{2}} & = & \mathbf{H}_h^{n+\frac{1}{2}} & - \quad \Delta t\, \mathbb{M}_\mu^{-1}\, \mathbb{G}\, \mathbf{E}_h^{n+1}, \\ \mathbf{E}_h^{n+1} & = & \mathbf{E}_h^{n} & + \quad \Delta t\, \mathbb{M}_\varepsilon^{-1}\, \mathbb{F}\, \mathbf{H}_h^{n+\frac{1}{2}}, \end{array}\right.$$

où $\mathbf{E}_h^n$ et $\mathbf{H}_h^{n+\frac{1}{2}}$ sont les approximations respectives de $\mathbf{E}_h$ et $\mathbf{H}_h$ en $t_n = t_0 + n\Delta t$ et $t_{n+\frac{1}{2}} = t_0 + (n+\frac{1}{2})\Delta t$ où Δt est le pas de temps.

Le schéma saute-mouton du quatrième ordre. Les champs $\mathbf{H}_h$ et $\mathbf{E}_h$ sont de nouveau évalués sur une grille décalée en temps. Le schéma s'écrit :

$$\left\{\begin{array}{lcll} \mathbf{H}_h^{n+\frac{3}{2}} & = & \mathbf{H}_h^{n+\frac{1}{2}} & + \quad \Delta t \left(\mathbf{T}_{H1} + \dfrac{\Delta t^2}{24} \mathbf{T}_{H3} \right), \\ \mathbf{E}_h^{n+1} & = & \mathbf{E}_h^{n} & + \quad \Delta t \left(\mathbf{T}_{E1} + \dfrac{\Delta t^2}{24} \mathbf{T}_{E3} \right), \end{array}\right.$$

où :

$$\left\{\begin{array}{lcl} \mathbf{T}_{H1} & = & -\mathbb{M}_\mu^{-1}\, \mathbb{G}\, \mathbf{E}_h^{n+1}, \\ \mathbf{T}_{H2} & = & \mathbb{M}_\varepsilon^{-1}\, \mathbb{F}\, \mathbf{T}_{H1}, \\ \mathbf{T}_{H3} & = & -\mathbb{M}_\mu^{-1}\, \mathbb{G}\, \mathbf{T}_{H2}, \end{array}\right.$$

et :

$$\left\{\begin{array}{lcl} \mathbf{T}_{E1} & = & \mathbb{M}_\varepsilon^{-1}\, \mathbb{F}\, \mathbf{H}_h^{n+\frac{1}{2}}, \\ \mathbf{T}_{E2} & = & -\mathbb{M}_\mu^{-1}\, \mathbb{G}\, \mathbf{T}_{E1}, \\ \mathbf{T}_{E3} & = & \mathbb{M}_\varepsilon^{-1}\, \mathbb{F}\, \mathbf{T}_{E2}, \end{array}\right.$$

2.4.3.2 Schéma de type Runge-Kutta

Il existe plusieurs variantes de schémas de Runge-Kutta. Nous utilisons ici un schéma de Runge-Kutta totalement explicite et qui minimise le nombre de vecteurs intermédiaires. Le schéma de Runge-Kutta du quatrième ordre qui est utilisé dans cette étude s'écrit :

$$\left\{\begin{array}{lcll} \mathbf{E}_h^{(0)} & = & \mathbf{E}_h^{n} & \\ \mathbf{E}_h^{(l)} & = & \mathbf{E}_h^{(0)} + \dfrac{\Delta t}{(5-l)}\, \mathbb{M}_\varepsilon^{-1} \mathbb{F}\, \mathbf{E}_h^{(l-1)} & \quad l = 1, \ldots, 4 \\ \mathbf{E}_h^{n+1} & = & \mathbf{E}_h^{(4)} & \end{array}\right.$$

$$\left\{\begin{array}{lcll} \mathbf{H}_h^{(0)} & = & \mathbf{H}_h^{n} & \\ \mathbf{H}_h^{(l)} & = & \mathbf{H}_h^{(0)} - \dfrac{\Delta t}{(5-l)}\, \mathbb{M}_\mu^{-1} \mathbb{G}\, \mathbf{H}_h^{(l-1)} & \quad l = 1, \ldots, 4 \\ \mathbf{H}_h^{n+1} & = & \mathbf{H}_h^{(4)} & \end{array}\right.$$

2.4.4 Etude de stabilité numérique

Nous nous proposons ici d'évaluer numériquement les valeurs maximales autorisées du nombre CFL pour les schémas d'intégration en temps discutés à la sous-section 2.4.3. Pour cela, pour chaque schéma en temps, nous réalisons plusieurs simulations pour différents degrés d'interpolation. Le problème test sélectionné pour cette étude est la propagation du mode fondamental. Le domaine de calcul est l'intervalle [0,1] aux extrémités duquel on applique une condition aux limites de réflexion totale. La solution initiale est donnée par :

$$E(x,0) = \sin(\pi x) \quad \text{et} \quad H(x, \frac{\Delta t}{2}) = \cos(\pi x)\sin(\omega \frac{\Delta t}{2}),$$

pour les schémas d'intégration en temps LF2 et LF4 et :

$$E(x,0) = \sin(\pi x) \quad \text{et} \quad H(x,0) = 0,$$

pour le schéma d'intégration en temps RK4. La solution analytique est donnée par :

$$E(x,t) = \sin(\pi x)\cos(\omega t) \quad \text{et} \quad H(x,t) = \cos(\pi x)\sin(\omega(t + \frac{\Delta t}{2})),$$

pour les schémas d'intégration en temps LF2 et LF4 et :

$$E(x,t) = \sin(\pi x)\cos(\omega t) \quad \text{et} \quad H(x,t) = \cos(\pi x)\sin(\omega t),$$

pour le schéma d'intégration en temps RK4.

L'intervalle [0,1] est discrétisé en $N-1$ intervalles de taille uniforme $\Delta x = 1/N$ et les points de discrétisation sont définis par $x_i = i\Delta x$ pour $i = 0, \cdots, N$. Lors des simulations, on observe les évolutions temporelles de l'erreur en norme L^2 entre la solution analytique et la solution approchée, ainsi que l'évolution temporelle de l'énergie discrète. En particulier, pour ce problème test uniquement basé sur des conditions aux limites de réflexion totale, l'énergie électromagnétique continue est exactement conservée. Dans le cas discret, il existe une valeur maximale du pas de temps (*i.e.* du nombre CFL) qui garantit la conservation de l'énergie électromagnétique discrète. Ainsi par dichotomie, on peut déterminer la valeur limite du nombre CFL au delà de laquelle le schéma devient instable. Par exemple, sur la figure 2.10, on montre les évolutions temporelles de l'erreur L^2 et de l'énergie discrète pour le schéma RK4 et une méthode d'interpolation linéaire dans la méthode Galerkin discontinue. Les résultats obtenus sont indépendants du type de fonction de base et sont résumés pour toutes les situations considérées ici dans la table 2.4.2. On note que la condition CFL est plus élevée pour les schémas en temps d'ordre 4 (LF4 et RK4) que pour les schémas en temps d'ordre 2 (LF2). De plus, il y a un facteur multiplicatif constant proche de 2.8 entre les valeurs du nombre CFL pour ces deux ordres. Enfin, la condition CFL diminue lorsque le degré d'interpolation augmente.

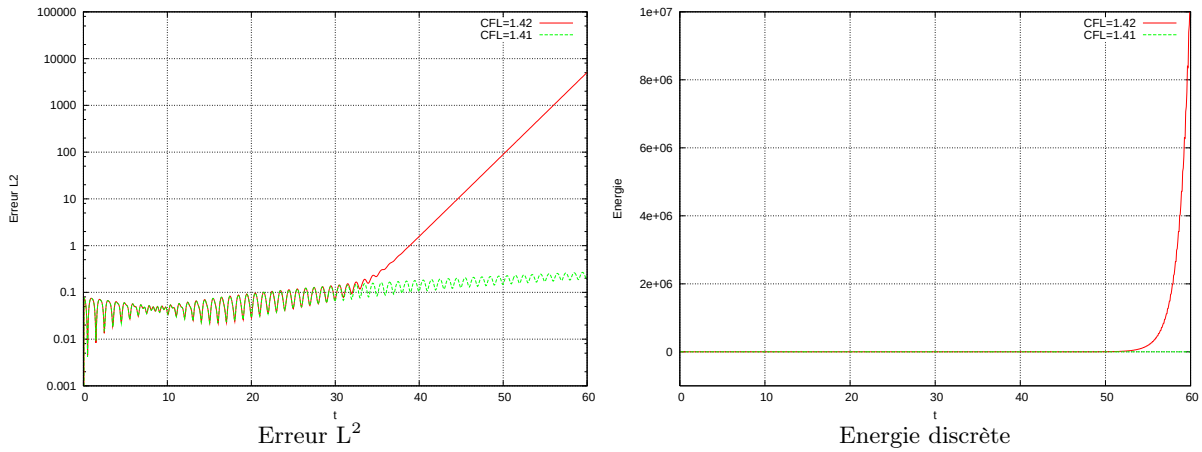
2.4.5 Propagation du mode fondamental

On reprend ici le problème test introduit dans la sous-section 2.4.4 que l'on simule maintenant en considérant différentes configurations de schémas en espace et en temps. Toutes les simulations ont été réalisées avec les paramètres suivants :

- l'évolution temporelle de la solution approchée est enregistrée au centre de l'intervalle *i.e.* $x_v = 0.5$,
- le temps final est fixé à $t_f = 2.10^{-7}$ s.

Pour ces simulations, les configurations sélectionnées diffèrent par :

- le schéma en temps (LF2, LF4 ou RK4),
- la méthode d'interpolation,
- le degré d'interpolation (de 1 à 5),



Erreur L^2 Energie discrète

Figure 2.10 – Evolutions temporelles de l'erreur L^2 et de l'énergie discrète.
RK4 - Interpolation $\mathbb{P}_1$.

Degré d'interpolation p	Schéma en temps	CFL maximum
1	LF2	0.50
1	LF4	1.42
1	RK4	1.41
2	LF2	0.24
2	LF4	0.70
2	RK4	0.70
3	LF2	0.15
3	LF4	0.42
3	RK4	0.42
4	LF2	0.10
4	LF4	0.28
4	RK4	0.28
5	LF2	0.07
5	LF4	0.20
5	RK4	0.20

Table 2.4.2 – Etude de stabilité numérique : valeurs maximales du nombre CFL.

- le nombre de points de discrétisation ($N = 11$, 21 ou 41),
- la condition CFL (on utilise les valeurs de la table 2.4.2).

Les figures 2.11 à 2.13 (distribution spatiale du champ électrique au temps final t_f) nous montrent que dans le cas d'une interpolation linéaire et pour une méthode d'interpolation de Lagrange, l'amplitude du mode fondamental est sous-estimée quelque soit le schéma d'intégration numérique utilisé mais que cette erreur tend à disparaître lorsque le nombre de points de discrétisation augmente.

Dans le but de comparer les différentes formes d'approximation polynomiale (Lagrange, Bernstein, Legendre, Canonique et Taylor), on évalue l'erreur L^2 entre la solution analytique et la solution numérique pour chacune des bases, pour un ordre d'interpolation allant de 1 à 5. On montre sur la figure 2.14 l'évolution temporelle de l'erreur L^2 pour la base de Lagrange. Des résultats très similaires sont obtenus pour les autres bases.

On a répertorié dans les tables 2.4.3 à 2.4.5 pour chaque schéma d'intégration en temps les valeurs maximales de l'erreur L^2 ainsi que les gains (taux de réduction de l'erreur) liés au raffinement du maillage.

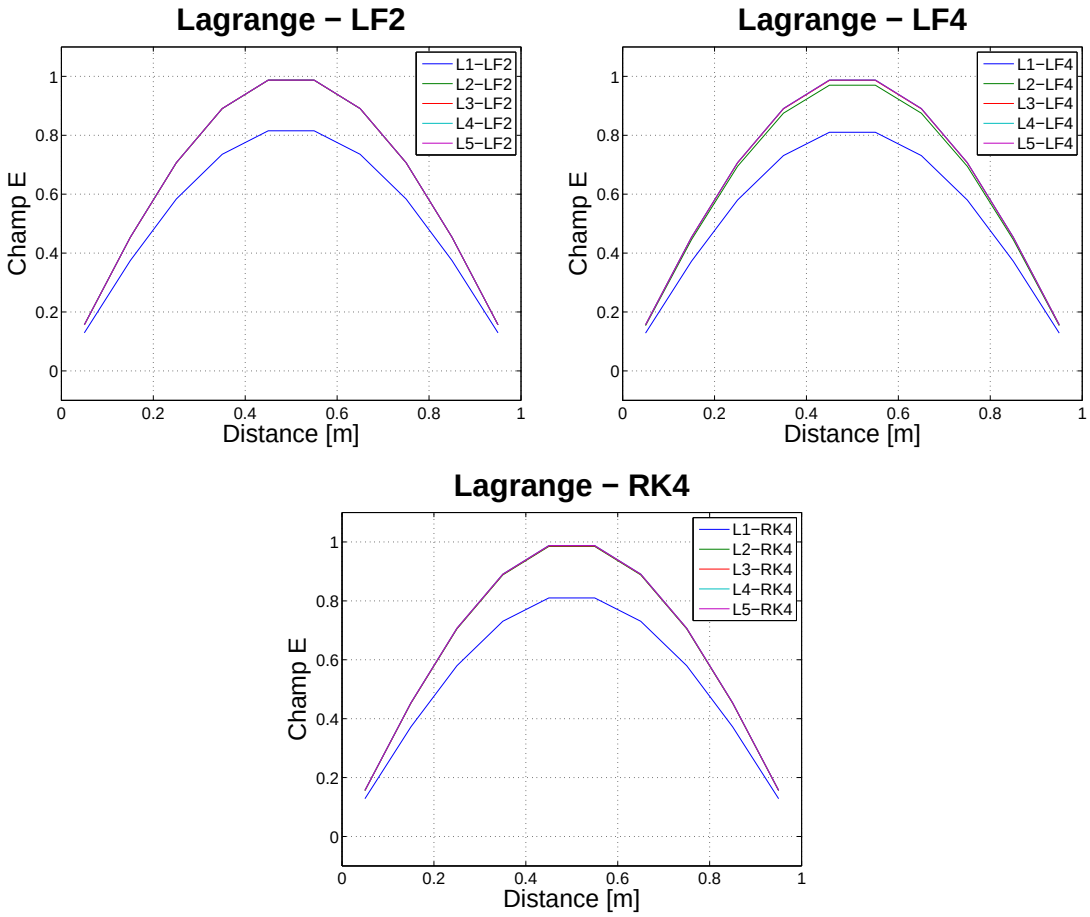


Figure 2.11 – Propagation du mode fondamental.
Champ électrique à $t = T_f$, N=11 points.

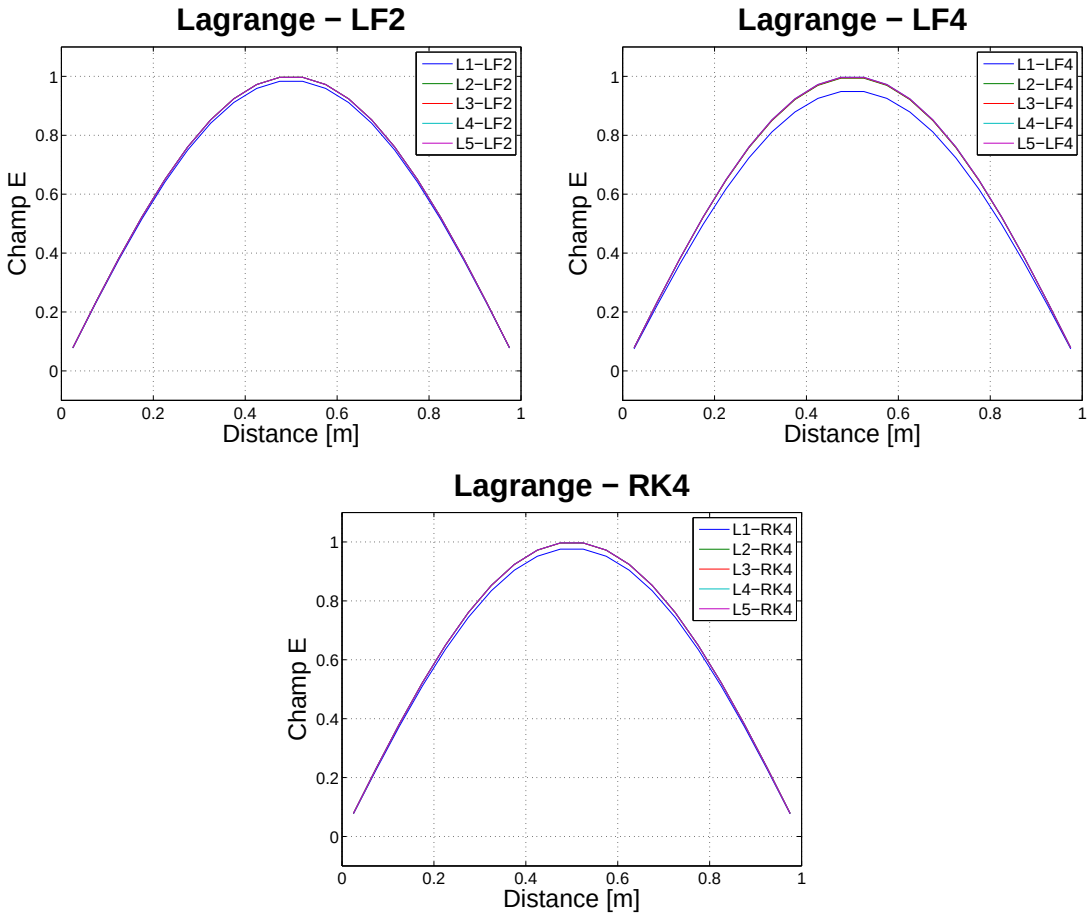


Figure 2.12 – Propagation du mode fondamental.
Champ électrique à $t = T_f$, N=21 points.

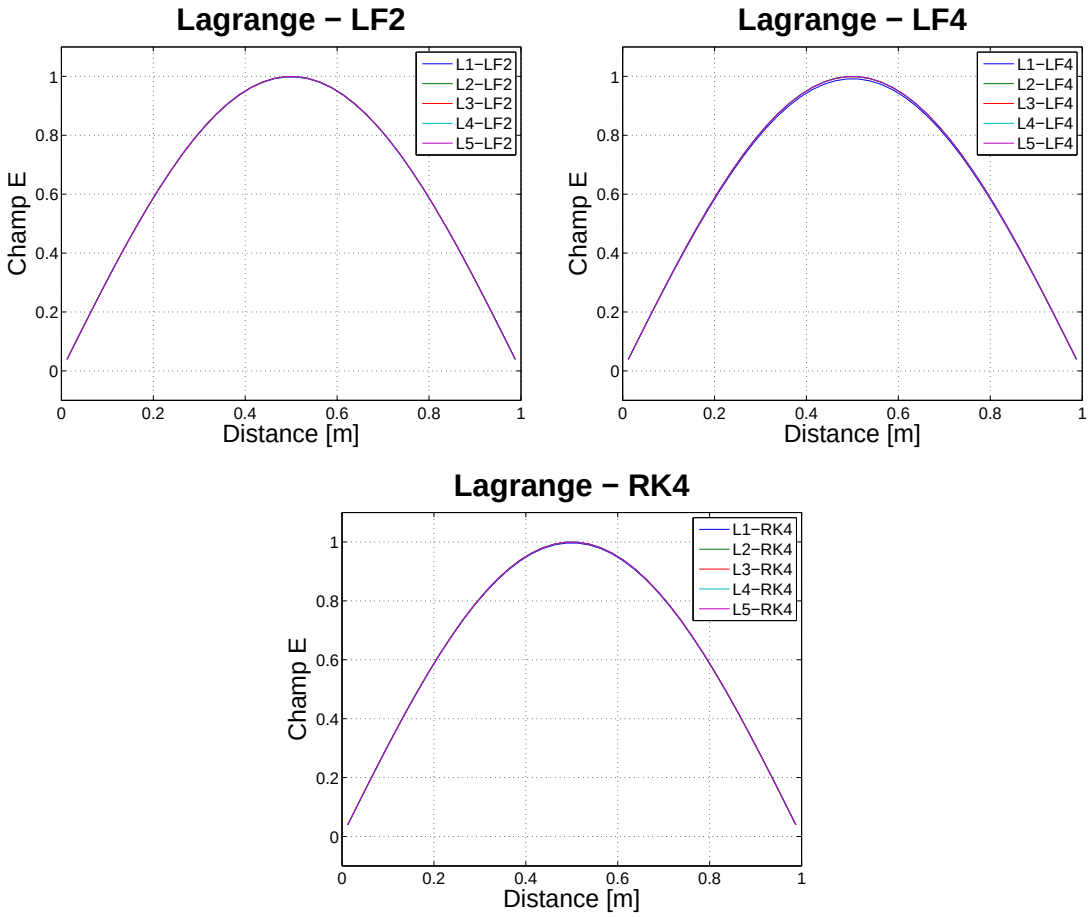


Figure 2.13 – Propagation du mode fondamental.
Champ électrique à $t = T_f$, N=41 points.

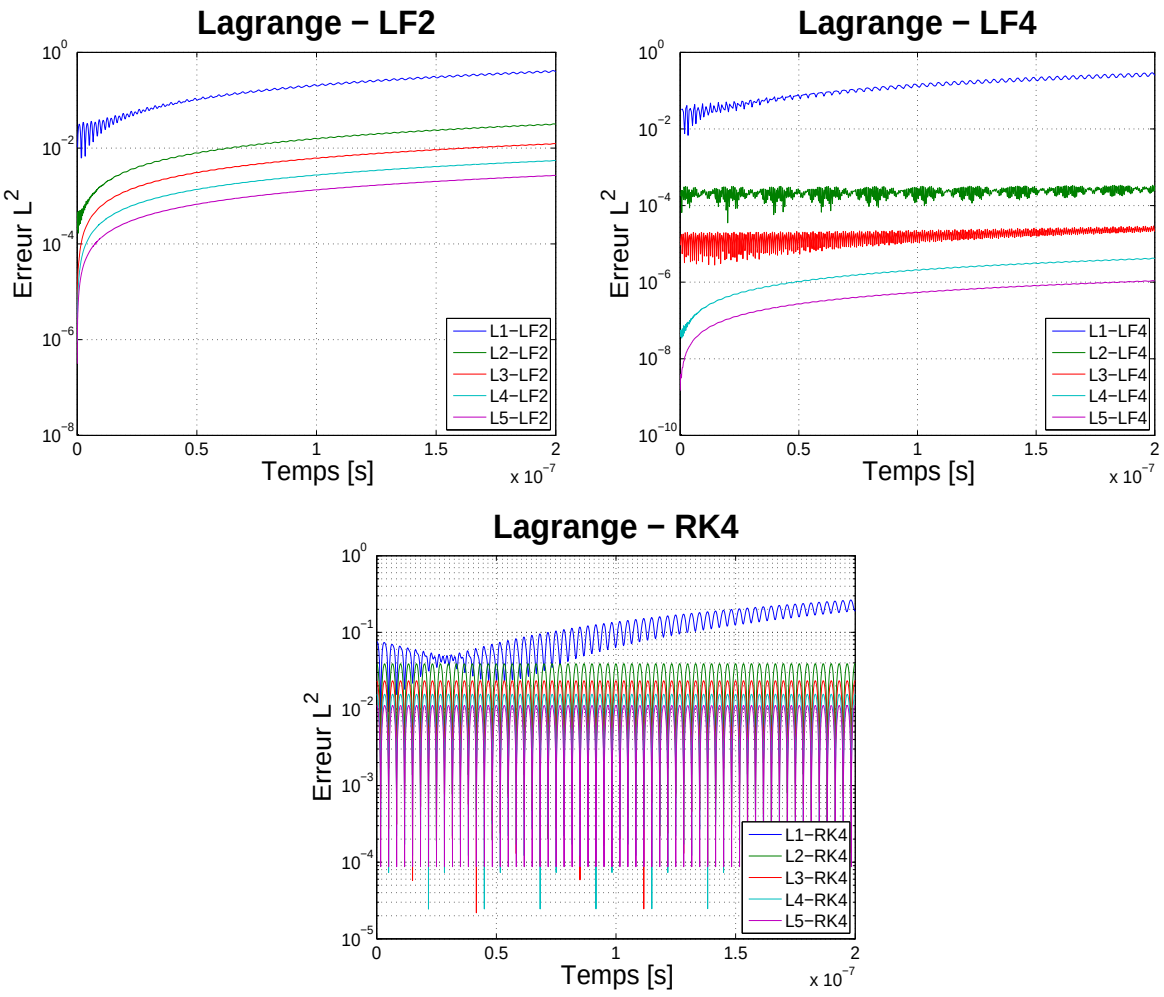


Figure 2.14 – Propagation du mode fondamental.
Evolution temporelle de l'erreur L^2, N=11 points.

Base	N=11 points	N=21 points	N=41 points	Gain 11 pts → 21 pts	Gain 11 pts → 41 pts
L1	$4.16 \cdot 10^{-1}$	$1.05 \cdot 10^{-1}$	$2.71 \cdot 10^{-2}$	3.96	15.35
B1	$4.19 \cdot 10^{-1}$	$1.055 \cdot 10^{-1}$	$2.71 \cdot 10^{-2}$	3.97	15.46
G1	$4.19 \cdot 10^{-1}$	$1.055 \cdot 10^{-1}$	$2.71 \cdot 10^{-2}$	3.97	15.46
C1	$4.19 \cdot 10^{-1}$	$1.055 \cdot 10^{-1}$	$2.71 \cdot 10^{-2}$	3.97	15.46
T1	$4.19 \cdot 10^{-1}$	$1.055 \cdot 10^{-1}$	$2.71 \cdot 10^{-2}$	3.97	15.46
L2	$3.20 \cdot 10^{-2}$	$7.94 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	4.03	16.16
B2	$3.20 \cdot 10^{-2}$	$7.94 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	4.03	16.16
G2	$3.20 \cdot 10^{-2}$	$7.94 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	4.03	16.16
C2	$3.20 \cdot 10^{-2}$	$7.94 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	4.03	16.16
T2	$3.20 \cdot 10^{-2}$	$7.94 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	4.03	16.16
L3	$1.24 \cdot 10^{-2}$	$3.09 \cdot 10^{-3}$	$7.71 \cdot 10^{-4}$	4.01	16.08
B3	$1.24 \cdot 10^{-2}$	$3.09 \cdot 10^{-3}$	$7.71 \cdot 10^{-4}$	4.01	16.08
G3	$1.24 \cdot 10^{-2}$	$3.09 \cdot 10^{-3}$	$7.71 \cdot 10^{-4}$	4.01	16.08
C3	$1.24 \cdot 10^{-2}$	$3.09 \cdot 10^{-3}$	$7.71 \cdot 10^{-4}$	4.01	16.08
T3	$1.24 \cdot 10^{-2}$	$3.09 \cdot 10^{-3}$	$7.71 \cdot 10^{-4}$	4.01	16.08
L4	$5.50 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	$3.43 \cdot 10^{-4}$	4.01	16.03
B4	$5.50 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	$3.43 \cdot 10^{-4}$	4.01	16.03
G4	$5.50 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	$3.43 \cdot 10^{-4}$	4.01	16.03
C4	$5.50 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	$3.43 \cdot 10^{-4}$	4.01	16.03
T4	$5.50 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	$3.43 \cdot 10^{-4}$	4.01	16.03
L5	$2.69 \cdot 10^{-3}$	$6.715 \cdot 10^{-4}$	$1.68 \cdot 10^{-4}$	4.01	16.01
B5	$2.69 \cdot 10^{-3}$	$6.71 \cdot 10^{-4}$	$1.68 \cdot 10^{-4}$	4.01	16.01
G5	$2.69 \cdot 10^{-3}$	$6.71 \cdot 10^{-4}$	$1.68 \cdot 10^{-4}$	4.01	16.01
C5	$2.69 \cdot 10^{-3}$	$6.71 \cdot 10^{-4}$	$1.68 \cdot 10^{-4}$	4.01	16.01
T5	$2.69 \cdot 10^{-3}$	$6.71 \cdot 10^{-4}$	$1.68 \cdot 10^{-4}$	4.01	16.01

TABLE 2.4.3 – Propagation du mode fondamental.
Valeurs maximales de l'erreur L^2, schéma LF2.

Bien que moins perceptible pour les schémas d'intégration en temps LF2 et RK4, ces résultats soulignent l'importance du raffinement du maillage sur la qualité des résultats et également l'influence de l'enrichissement de l'ordre d'approximation sur la qualité de l'interpolation. Cela est encore une fois particulièrement visible dans le cas du schéma d'intégration en temps LF4 (voir la table 2.4.4) où l'on voit que l'erreur maximale atteint des valeurs très petites à partir de l'ordre 2 et que d'une manière générale, l'enrichissement de l'ordre d'interpolation combiné à une discrétisation plus fine du domaine génèrent des gains considérables pour toutes les fonctions de base choisies.

Les dernières tables 2.4.6 à 2.4.8 ne font que confirmer de manière encore plus nette l'avantage à choisir le schéma d'intégration en temps LF4 comparativement aux autres schémas pour le problème test considéré. On relève encore que les solutions approchées les plus précises sont obtenues pour les degrés d'interpolation 2 à 5 et que l'interpolation linéaire donnant les erreurs les plus importantes ne permet pas de différentier de manière distincte les schémas d'intégration en temps. On peut également ajouter que pour un degré d'interpolation au moins égal à 2, le schéma RK4 s'avère moins précis que le schéma LF2.

En conclusion de cette série de résultats numériques, nous retiendrons que le type d'interpolation polynomiale affecte peu la précision des solutions approchées.

Base	N=11 points	N=21 points	N=41 points	Gain 11 pts → 21 pts	Gain 11 pts → 41 pts
L1	$2.93 \cdot 10^{-1}$	$7.23 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	4.05	15.10
B1	$2.95 \cdot 10^{-1}$	$7.24 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	4.07	15.21
G1	$2.95 \cdot 10^{-1}$	$7.24 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	4.07	15.21
C1	$2.95 \cdot 10^{-1}$	$7.24 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	4.07	15.21
T1	$2.95 \cdot 10^{-1}$	$7.24 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	4.07	15.21
L2	$3.41 \cdot 10^{-4}$	$3.49 \cdot 10^{-5}$	$4.07 \cdot 10^{-6}$	9.77	83.78
B2	$1.82 \cdot 10^{-4}$	$1.06 \cdot 10^{-5}$	$6.49 \cdot 10^{-7}$	17.17	280.43
G2	$1.82 \cdot 10^{-4}$	$1.06 \cdot 10^{-5}$	$6.49 \cdot 10^{-7}$	17.17	280.43
C2	$1.82 \cdot 10^{-4}$	$1.06 \cdot 10^{-5}$	$6.49 \cdot 10^{-7}$	17.17	280.43
T2	$1.82 \cdot 10^{-4}$	$1.06 \cdot 10^{-5}$	$6.49 \cdot 10^{-7}$	17.17	280.43
L3	$3.93 \cdot 10^{-5}$	$2.65 \cdot 10^{-6}$	$2.89 \cdot 10^{-7}$	14.83	135.99
B3	$2.94 \cdot 10^{-5}$	$2.65 \cdot 10^{-6}$	$2.90 \cdot 10^{-7}$	11.09	101.38
G3	$2.94 \cdot 10^{-5}$	$2.65 \cdot 10^{-6}$	$2.90 \cdot 10^{-7}$	11.09	101.38
C3	$2.94 \cdot 10^{-5}$	$2.65 \cdot 10^{-6}$	$2.90 \cdot 10^{-7}$	11.09	101.38
T3	$2.94 \cdot 10^{-5}$	$2.65 \cdot 10^{-6}$	$2.90 \cdot 10^{-7}$	11.09	101.38
L4	$4.23 \cdot 10^{-6}$	$2.62 \cdot 10^{-7}$	$1.63 \cdot 10^{-8}$	16.15	259.51
B4	$4.23 \cdot 10^{-6}$	$2.62 \cdot 10^{-7}$	$1.63 \cdot 10^{-8}$	16.15	259.51
G4	$4.23 \cdot 10^{-6}$	$2.62 \cdot 10^{-7}$	$1.63 \cdot 10^{-8}$	16.15	259.51
C4	$4.23 \cdot 10^{-6}$	$2.62 \cdot 10^{-7}$	$1.63 \cdot 10^{-8}$	16.15	259.51
T4	$4.23 \cdot 10^{-6}$	$2.62 \cdot 10^{-7}$	$1.63 \cdot 10^{-8}$	16.15	259.51
L5	$1.09 \cdot 10^{-6}$	$6.79 \cdot 10^{-8}$	$4.26 \cdot 10^{-9}$	16.05	255.87
B5	$1.09 \cdot 10^{-6}$	$6.79 \cdot 10^{-8}$	$4.26 \cdot 10^{-9}$	16.05	255.87
G5	$1.09 \cdot 10^{-6}$	$6.79 \cdot 10^{-8}$	$4.26 \cdot 10^{-9}$	16.05	255.87
C5	$1.09 \cdot 10^{-6}$	$6.79 \cdot 10^{-8}$	$4.26 \cdot 10^{-9}$	16.05	255.87
T5	$1.09 \cdot 10^{-6}$	$6.79 \cdot 10^{-8}$	$4.26 \cdot 10^{-9}$	16.05	255.87

TABLE 2.4.4 – Propagation du mode fondamental.
Valeurs maximales de l'erreur L^2, schéma LF4.

Base	N=11 points	N=21 points	N=41 points	Gain 11 pts → 21 pts	Gain 11 pts → 41 pts
L1	$2.65 \cdot 10^{-1}$	$6.87 \cdot 10^{-2}$	$1.95 \cdot 10^{-2}$	3.86	13.59
B1	$2.67 \cdot 10^{-1}$	$6.88 \cdot 10^{-2}$	$1.95 \cdot 10^{-2}$	3.88	13.69
G1	$2.67 \cdot 10^{-1}$	$6.88 \cdot 10^{-2}$	$1.95 \cdot 10^{-2}$	3.88	13.69
C1	$2.67 \cdot 10^{-1}$	$6.88 \cdot 10^{-2}$	$1.95 \cdot 10^{-2}$	3.88	13.69
T1	$2.67 \cdot 10^{-1}$	$6.88 \cdot 10^{-2}$	$1.95 \cdot 10^{-2}$	3.88	13.69
L2	$3.90 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	$9.72 \cdot 10^{-3}$	2.01	4.01
B2	$3.90 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	$9.72 \cdot 10^{-3}$	2.01	4.01
G2	$3.90 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	$9.72 \cdot 10^{-3}$	2.01	4.01
C2	$3.90 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	$9.72 \cdot 10^{-3}$	2.01	4.01
T2	$3.90 \cdot 10^{-2}$	$1.94 \cdot 10^{-2}$	$9.72 \cdot 10^{-3}$	2.01	4.01
L3	$2.33 \cdot 10^{-2}$	$1.17 \cdot 10^{-2}$	$5.83 \cdot 10^{-3}$	1.99	4.00
B3	$2.33 \cdot 10^{-2}$	$1.17 \cdot 10^{-2}$	$5.83 \cdot 10^{-3}$	1.99	4.00
G3	$2.33 \cdot 10^{-2}$	$1.17 \cdot 10^{-2}$	$5.83 \cdot 10^{-3}$	1.99	4.00
C3	$2.33 \cdot 10^{-2}$	$1.17 \cdot 10^{-2}$	$5.83 \cdot 10^{-3}$	1.99	4.00
T3	$2.33 \cdot 10^{-2}$	$1.17 \cdot 10^{-2}$	$5.83 \cdot 10^{-3}$	1.99	4.00
L4	$1.56 \cdot 10^{-2}$	$7.78 \cdot 10^{-3}$	$3.89 \cdot 10^{-3}$	2.01	4.01
B4	$1.56 \cdot 10^{-2}$	$7.78 \cdot 10^{-3}$	$3.89 \cdot 10^{-3}$	2.01	4.01
G4	$1.56 \cdot 10^{-2}$	$7.78 \cdot 10^{-3}$	$3.89 \cdot 10^{-3}$	2.01	4.01
C4	$1.56 \cdot 10^{-2}$	$7.78 \cdot 10^{-3}$	$3.89 \cdot 10^{-3}$	2.01	4.01
T4	$1.56 \cdot 10^{-2}$	$7.78 \cdot 10^{-3}$	$3.89 \cdot 10^{-3}$	2.01	4.01
L5	$1.11 \cdot 10^{-2}$	$5.55 \cdot 10^{-3}$	$2.78 \cdot 10^{-3}$	2.00	3.99
B5	$1.11 \cdot 10^{-2}$	$5.55 \cdot 10^{-3}$	$2.78 \cdot 10^{-3}$	2.00	3.99
G5	$1.11 \cdot 10^{-2}$	$5.55 \cdot 10^{-3}$	$2.78 \cdot 10^{-3}$	2.00	3.99
C5	$1.11 \cdot 10^{-2}$	$5.55 \cdot 10^{-3}$	$2.78 \cdot 10^{-3}$	2.00	3.99
T5	$1.11 \cdot 10^{-2}$	$5.55 \cdot 10^{-3}$	$2.78 \cdot 10^{-3}$	2.00	3.99

TABLE 2.4.5 – Propagation du mode fondamental.
Valeurs maximales de l'erreur L^2, schéma RK4.

Base	Gain LF2 → LF4	Gain LF2 → RK4	Gain LF4 → RK4
L1	1.42	1.57	1.11
B1	1.42	1.57	1.10
G1	1.42	1.57	1.10
C1	1.42	1.57	1.10
T1	1.42	1.57	1.10
L2	93.84	0.82	0.009
B2	175.82	0.82	0.005
G2	175.82	0.82	0.005
C2	175.82	0.82	0.005
T2	175.82	0.82	0.005
L3	315.52	0.53	0.002
B3	421.77	0.53	0.001
G3	421.77	0.53	0.001
C3	421.77	0.53	0.001
T3	421.77	0.53	0.001
L4	1300.24	0.35	0.0003
B4	1300.24	0.35	0.0003
G4	1300.24	0.35	0.0003
C4	1300.24	0.35	0.0003
T4	1300.24	0.35	0.0003
L5	2467.89	0.24	0.0001
B5	2467.89	0.24	0.0001
G5	2467.89	0.24	0.0001
C5	2467.89	0.24	0.0001
T5	2467.89	0.24	0.0001

TABLE 2.4.6 – Propagation du mode fondamental.
Comparatif des gains obtenus sur l'erreur maximale L^2
entre les différents schémas en temps pour N=11 points.

Base	Gain LF2 → LF4	Gain LF2 → RK4	Gain LF4 → RK4
L1	1.45	1.53	1.06
B1	1.46	1.53	1.06
G1	1.46	1.53	1.06
C1	1.46	1.53	1.06
T1	1.46	1.53	1.06
L2	227.51	0.41	0.002
B2	749.06	0.41	0.0005
G2	749.06	0.41	0.0005
C2	749.06	0.41	0.0005
T2	749.06	0.41	0.0005
L3	1166.04	0.26	0.0002
B3	1166.04	0.26	0.0002
G3	1166.04	0.26	0.0002
C3	1166.04	0.26	0.0002
T3	1166.04	0.26	0.0002
L4	5229.01	0.18	0.00003
B4	5229.01	0.18	0.00003
G4	5229.01	0.18	0.00003
C4	5229.01	0.18	0.00003
T4	5229.01	0.18	0.00003
L5	9889.54	0.12	0.00001
B5	9882.18	0.12	0.00001
G5	9882.18	0.12	0.00001
C5	9882.18	0.12	0.00001
T5	9882.18	0.12	0.00001

TABLE 2.4.7 – Propagation du mode fondamental.
Comparatif des gains obtenus sur l'erreur maximale L^2
entre les différents schémas en temps pour N=21 points.

Base	Gain LF2 → LF4	Gain LF2 → RK4	Gain LF4 → RK4
L1	1.40	1.39	0.99
B1	1.40	1.39	0.99
G1	1.40	1.39	0.99
C1	1.40	1.39	0.99
T1	1.40	1.39	0.99
L2	486.49	0.20	0.0004
B2	3050.85	0.20	0.00007
G2	3050.85	0.20	0.00007
C2	3050.85	0.20	0.00007
T2	3050.85	0.20	0.00007
L3	2667.82	0.13	0.00005
B3	2658.62	0.13	0.00005
G3	2658.62	0.13	0.00005
C3	2658.62	0.13	0.00005
T3	2658.62	0.13	0.00005
L4	21042.94	0.09	0.000004
B4	21042.94	0.09	0.000004
G4	21042.94	0.09	0.000004
C4	21042.94	0.09	0.000004
T4	21042.94	0.09	0.000004
L5	39436.62	0.06	0.000002
B5	39436.62	0.06	0.000002
G5	39436.62	0.06	0.000002
C5	39436.62	0.06	0.000002
T5	39436.62	0.06	0.000002

TABLE 2.4.8 – Propagation du mode fondamental.
Comparatif des gains obtenus sur l'erreur maximale L^2
entre les différents schémas en temps pour N=41 points.

2.4.6 Propagation d'un pulse

On étudie maintenant la propagation d'un pulse dans un domaine ouvert. Le domaine de calcul est ici l'intervalle [0,4] aux extrémités duquel est imposée une condition absorbante. On considère deux types de pulse : un pulse gaussien et un pulse triangulaire. Comme pour le problème test du mode fondamental, l'intervalle [0,4] est discrétisé en $N-1$ intervalles de taille uniforme $\Delta x = 1/N$ et les points de discrétisation sont définis par $x_i = i\Delta x$ pour $i = 0, \cdots, N$.

Toutes les simulations ont été réalisées avec les paramètres suivants :

- le point de visualisation de l'évolution temporelle de la solution approchée est situé en $x_v = 1.5$,
- le temps final est fixé à $t_f = 0.333.10^{-8}$ s,
- pour chacun des pulses étudiés, le signal de base est centré en $x = 1$,
- le nombre de points de discrétisation est successivement fixé à $N = 81$, $N = 121$ et $N = 201$.

2.4.6.1 Pulse gaussien

On s'intéresse dans un premier temps à la propagation du pulse gaussien. Sur les figures 2.15 à 2.17 on visualise la distribution spatiale du champ électrique au temps final pour différents nombres de points de discrétisation et des simulations basées sur la base de Lagrange. Des résultats très similaires sont obtenus pour les autres bases. On remarque que l'amplitude du pulse est correcte dans chaque situation mais que des différences existent tout de même. Pour mieux appréhender celles-ci, des zooms sont représentés sur les figures 2.18 à 2.20, qui concernent la partie inférieure gauche du pulse. Ces figures illustrent des résultats intuitifs : plus l'ordre d'interpolation est faible et plus la courbe finale s'écarte de sa valeur théorique. Par ailleurs, ce défaut sur la courbe n'est presque pas visible lorsque l'on utilise un schéma d'intégration en temps RK4, seul l'ordre 1 fait apparaitre une différence. Les zooms effectués sur le sommet du pulse gaussien (voir les figures 2.21 à 2.23) ne permettent pas de voir l'influence du p-enrichissement mais nous montrent que le décalage des courbes correspondant à chaque ordre est le plus important pour le schéma RK4 et que ces écarts s'amménuisent lorsque que l'on raffine le maillage.

L'évolution temporelle de l'erreur L^2 pour la base de Lagrange et pour chacun des schémas en temps est montrée sur la figure 2.24 pour le maillage à 81 points. On voit là que le schéma RK4 donne les résultats les plus singuliers particulièrement en ce qui concerne les ordres d'interpolation 1 et 2.

Les tableaux 2.4.9 et 2.4.10 nous indiquent que les schémas en temps LF2 et LF4 semblent ne pas se dinstinguer réellement et présentent tous deux la même régularité au niveau des gains, aucune base ne semble mieux adaptée qu'une autre. Les valeurs du tableau 2.4.11 montrent en revanche que le gain diminue lorsque le degré d'interpolation augmente. On remarque aussi que dans le cas d'un schéma en temps RK4 et en isolant le cas particulier de l'interpolation linéaire, le fait de raffiner la discrétisation du domaine n'améliore pas ou peu la précision des résultats.

Si l'on se penche sur le tableau 2.4.12 on retrouve un écart considérable entre les résultats d'une interpolation linéaire et ceux d'une interpolation d'ordre plus élevé. Ce tableau atteste la supériorité du schéma RK4 qui se distingue très nettement des autres schémas et pour lequel le facteur d'amplification le plus important est atteint pour une interpolation quadratique.

2.4.6.2 Pulse triangulaire

La distribution spatiale du champ électrique au temps final correspondant à une approximation numérique utilisant la base de Lagrange et pour différents nombres de points de discrétisation est visualisée sur les figures 2.25 à 2.27. La première remarque concerne l'amplitude du signal, l'évaluation du sommet étant délicate et imprécise, une discrétisation minimale du domaine est requise afin de s'assurer d'une approximation correcte de la valeur maximale. Ensuite, comme nous l'avions vu dans le cas du pulse gaussien, des irrégularités apparaissent à la base du pulse mais cette fois ci, le phénomène est plus marqué pour le schéma en temps LF2 et tend à se dissiper lorsque le maillage se raffine.

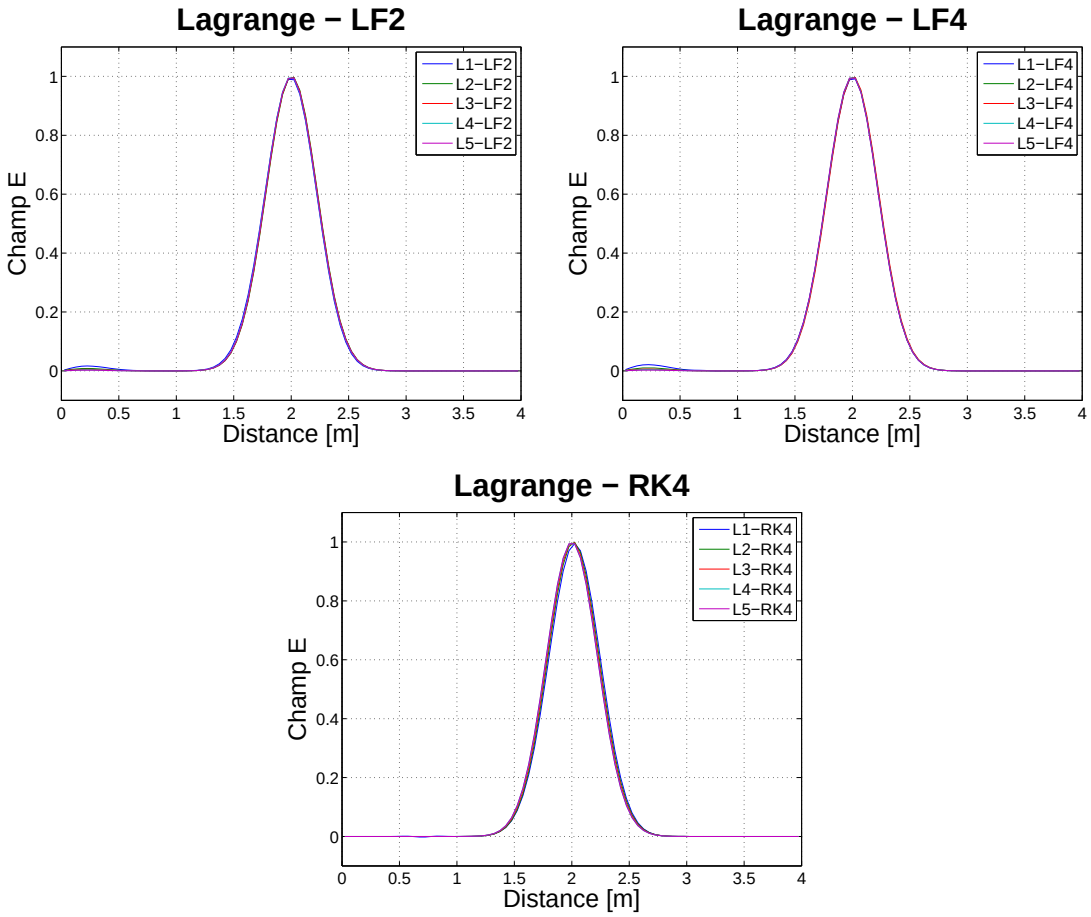


Figure 2.15 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=81 points.

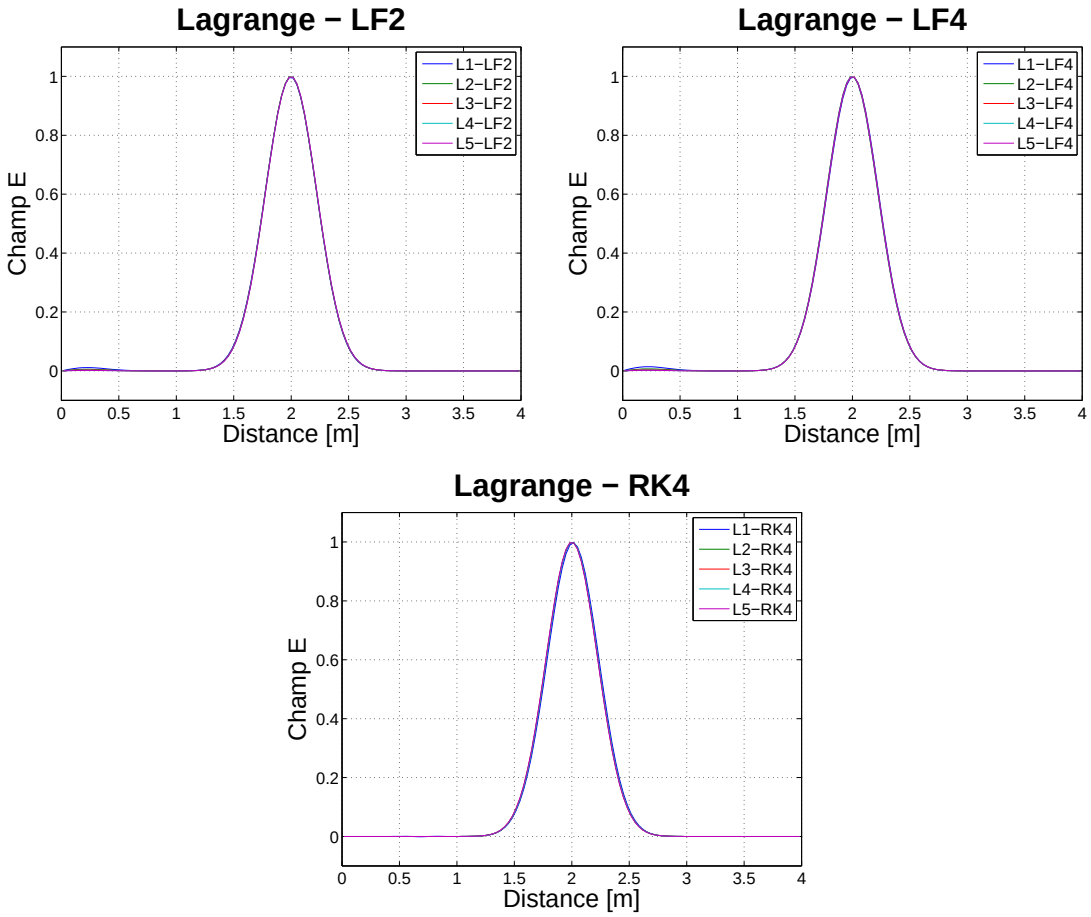


Figure 2.16 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=121 points.

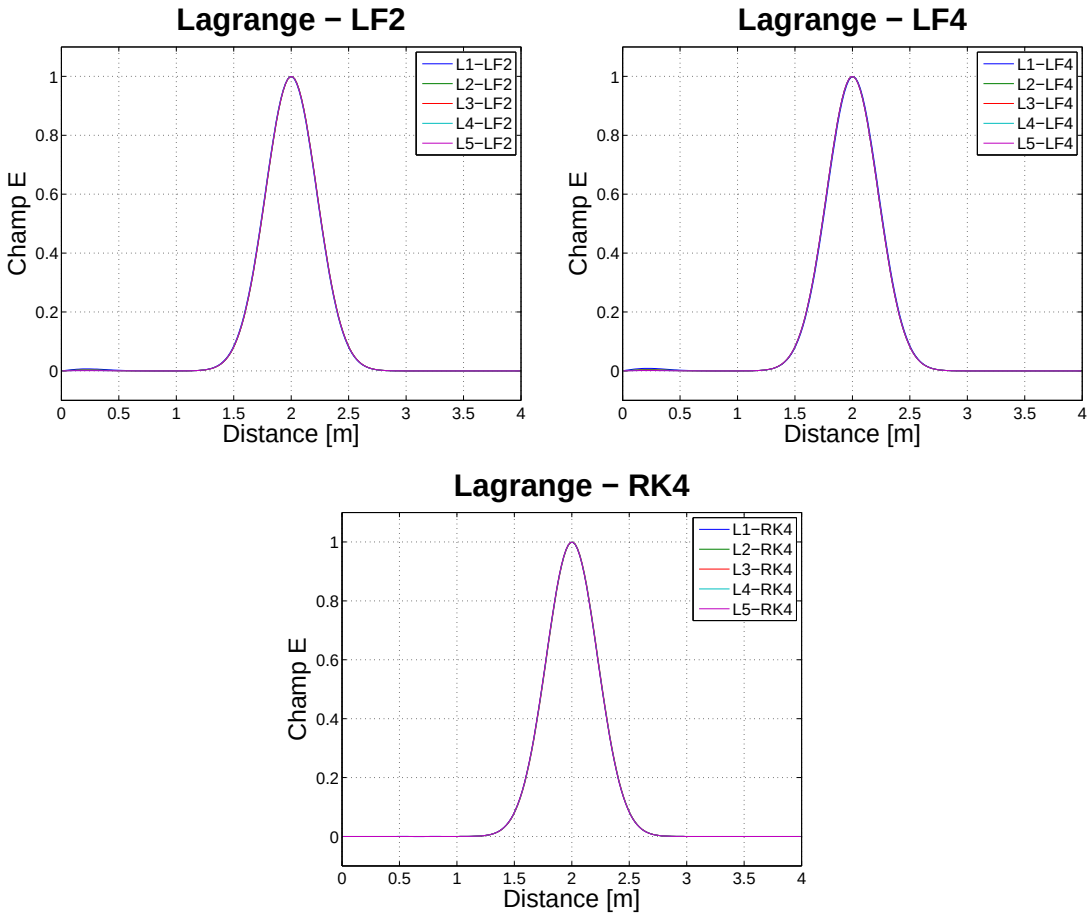


Figure 2.17 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=201 points.

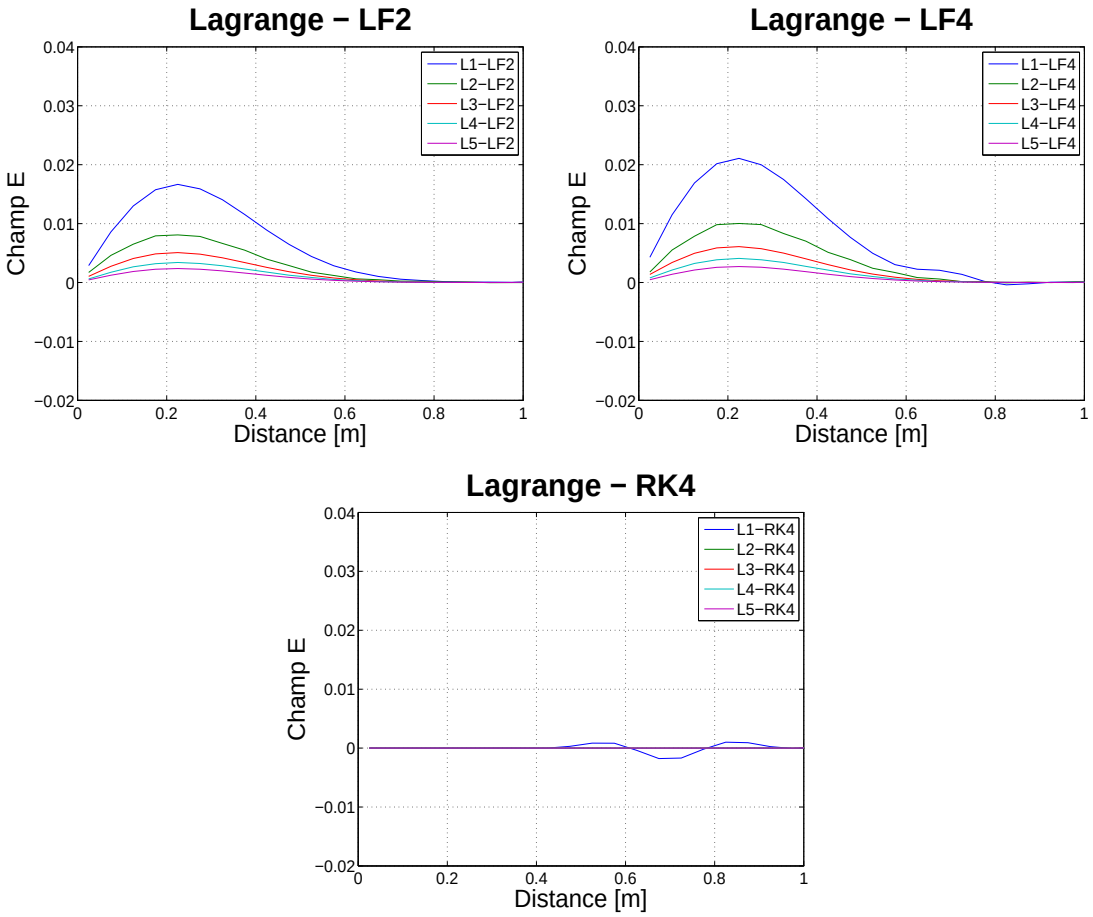


Figure 2.18 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=81 points.

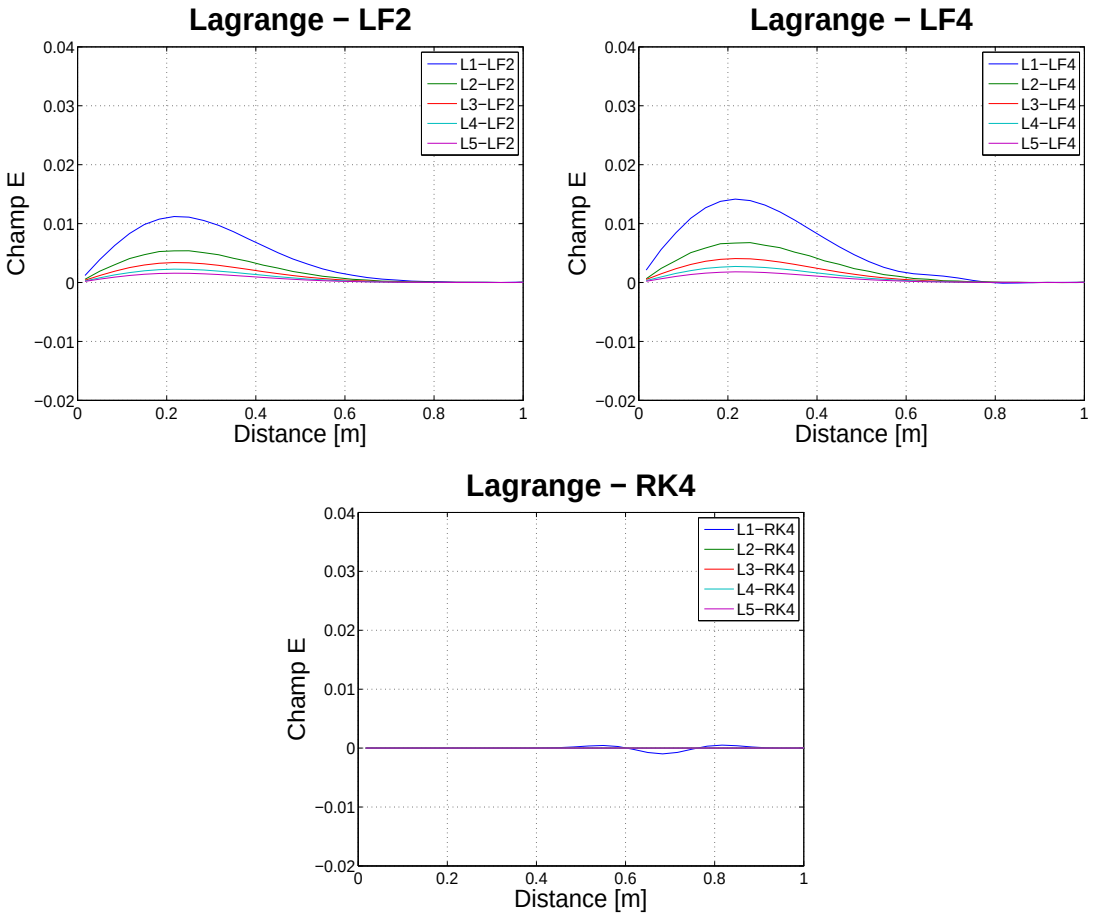


Figure 2.19 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=121 points.

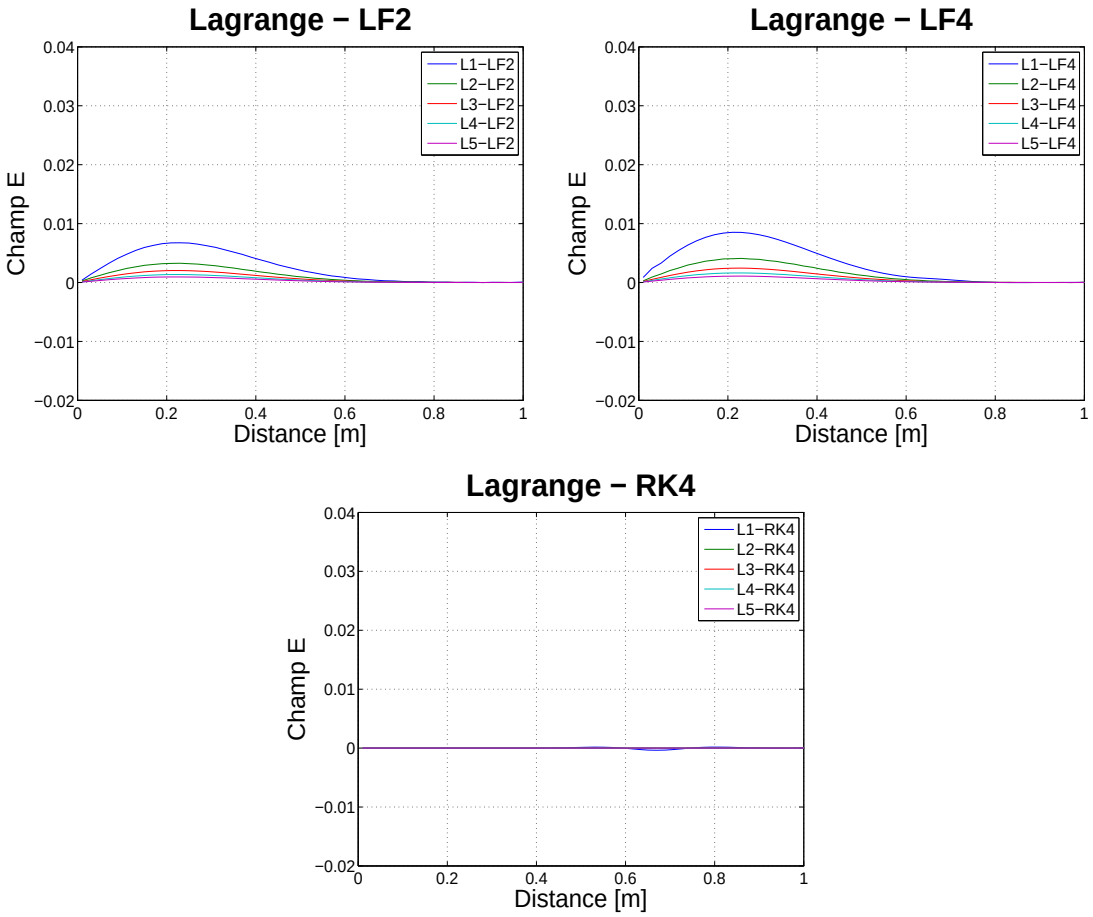


Figure 2.20 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=201 points.

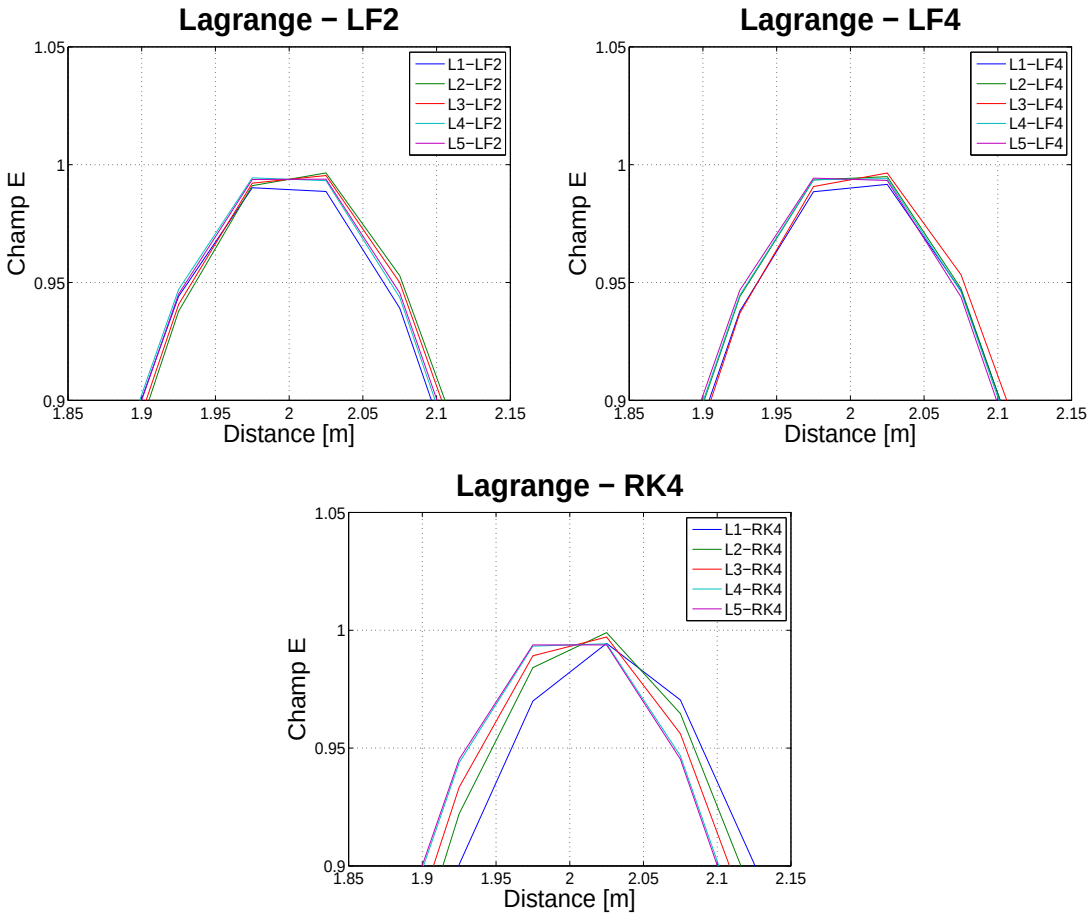


Figure 2.21 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=81 points.

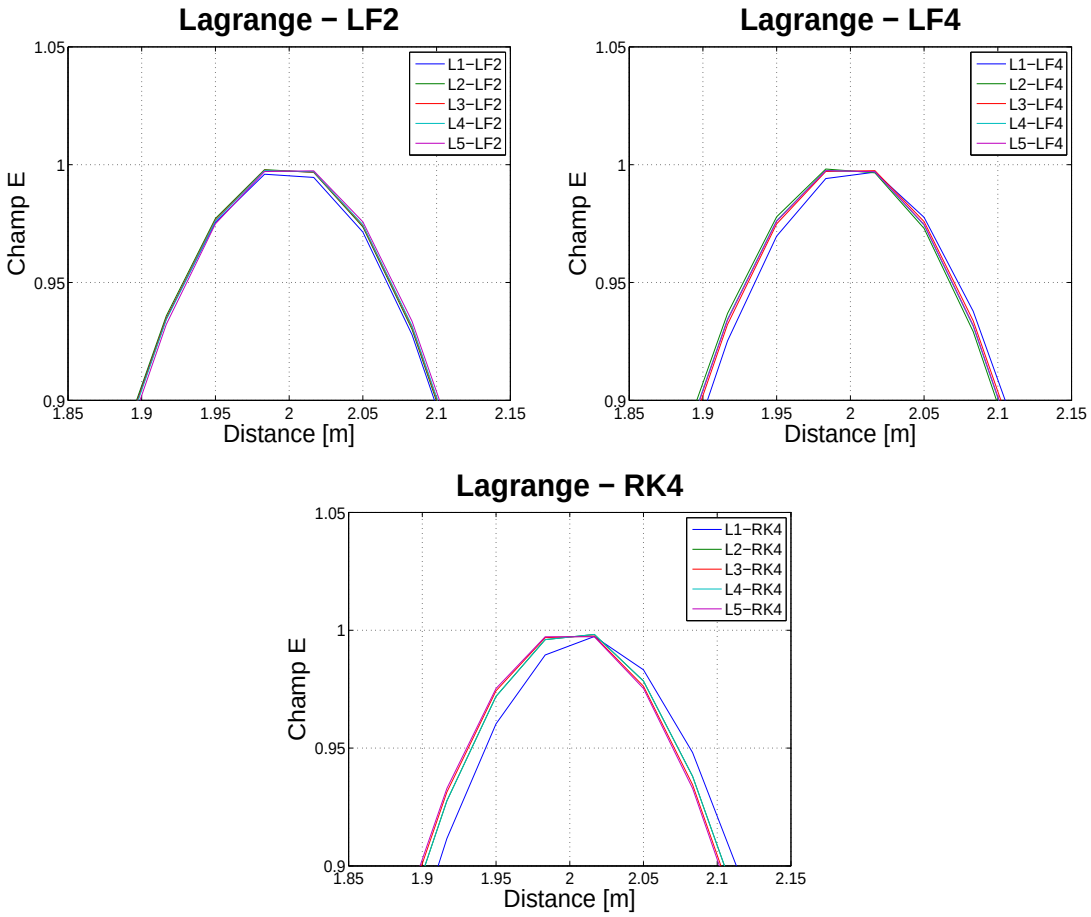


Figure 2.22 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=121 points.

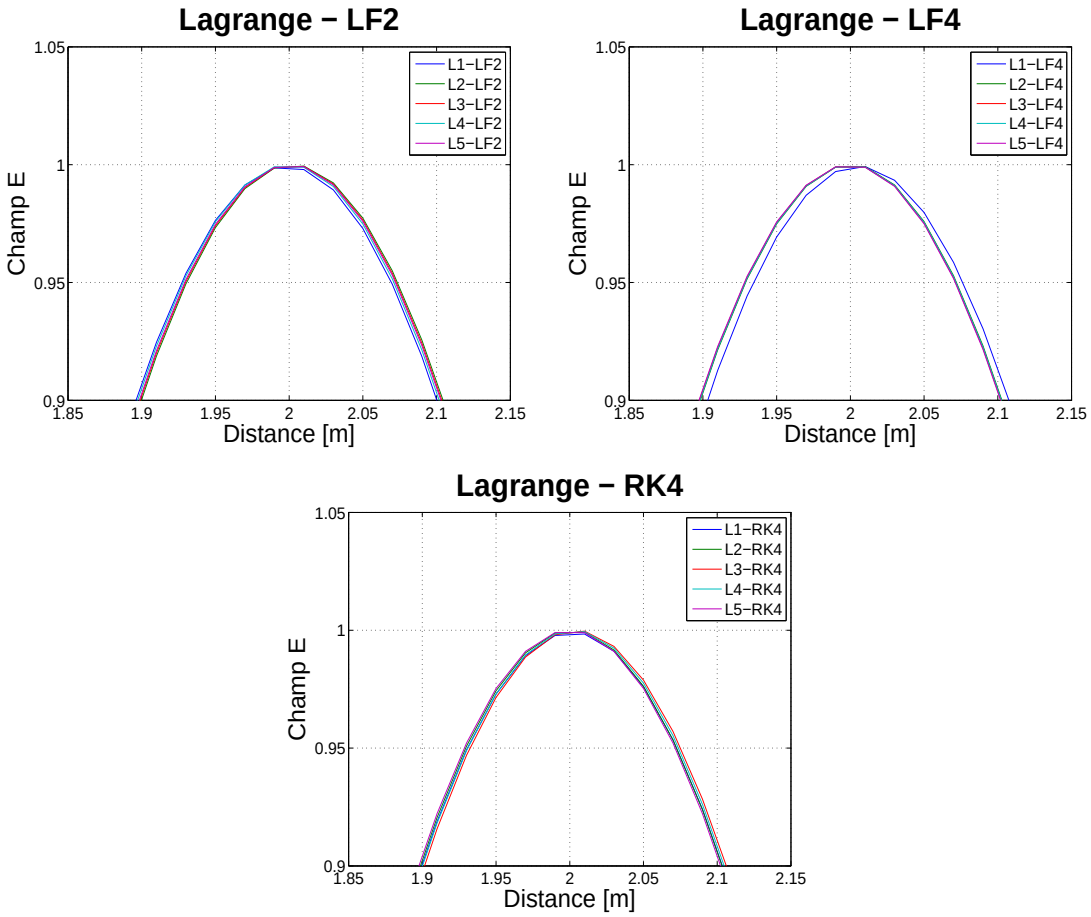


Figure 2.23 – Propagation du pulse gaussien.
Champ électrique à $t = T_f$, N=201 points.

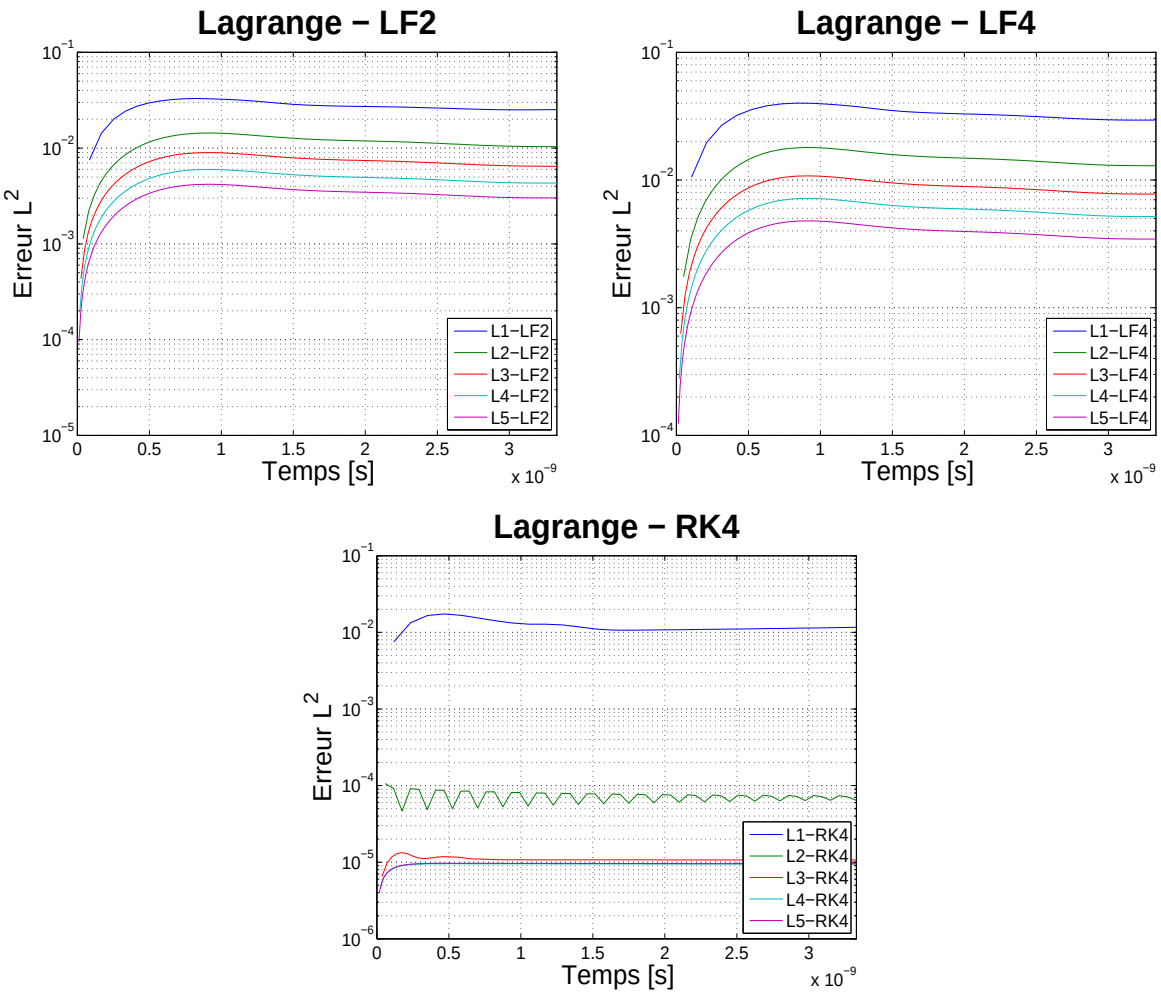


Figure 2.24 – Propagation du pulse gaussien.
Evolution temporelle de l'erreur L^2, N=81 points.

Base	N=81 points	N=121 points	N=201 points	Gain 81 pts → 121 pts	Gain 81 pts → 201 pts
L1	$3.29 \cdot 10^{-2}$	$2.19 \cdot 10^{-2}$	$1.31 \cdot 10^{-2}$	1.50	2.51
B1	$3.30 \cdot 10^{-2}$	$2.20 \cdot 10^{-2}$	$1.32 \cdot 10^{-2}$	1.50	2.51
G1	$3.30 \cdot 10^{-2}$	$2.20 \cdot 10^{-2}$	$1.32 \cdot 10^{-2}$	1.50	2.51
C1	$3.30 \cdot 10^{-2}$	$2.20 \cdot 10^{-2}$	$1.32 \cdot 10^{-2}$	1.50	2.51
T1	$3.30 \cdot 10^{-2}$	$2.20 \cdot 10^{-2}$	$1.32 \cdot 10^{-2}$	1.50	2.51
L2	$1.44 \cdot 10^{-2}$	$9.58 \cdot 10^{-3}$	$5.75 \cdot 10^{-3}$	1.50	2.50
B2	$1.44 \cdot 10^{-2}$	$9.58 \cdot 10^{-3}$	$5.75 \cdot 10^{-3}$	1.50	2.50
G2	$1.44 \cdot 10^{-2}$	$9.58 \cdot 10^{-3}$	$5.75 \cdot 10^{-3}$	1.50	2.50
C2	$1.44 \cdot 10^{-2}$	$9.58 \cdot 10^{-3}$	$5.75 \cdot 10^{-3}$	1.50	2.50
T2	$1.44 \cdot 10^{-2}$	$9.58 \cdot 10^{-3}$	$5.75 \cdot 10^{-3}$	1.50	2.50
L3	$8.98 \cdot 10^{-3}$	$5.99 \cdot 10^{-3}$	$3.59 \cdot 10^{-3}$	1.50	2.50
B3	$8.98 \cdot 10^{-3}$	$5.99 \cdot 10^{-3}$	$3.59 \cdot 10^{-3}$	1.50	2.50
G3	$8.98 \cdot 10^{-3}$	$5.99 \cdot 10^{-3}$	$3.59 \cdot 10^{-3}$	1.50	2.50
C3	$8.98 \cdot 10^{-3}$	$5.99 \cdot 10^{-3}$	$3.59 \cdot 10^{-3}$	1.50	2.50
T3	$8.98 \cdot 10^{-3}$	$5.99 \cdot 10^{-3}$	$3.59 \cdot 10^{-3}$	1.50	2.50
L4	$5.99 \cdot 10^{-3}$	$3.99 \cdot 10^{-3}$	$2.39 \cdot 10^{-3}$	1.50	2.51
B4	$5.99 \cdot 10^{-3}$	$3.99 \cdot 10^{-3}$	$2.39 \cdot 10^{-3}$	1.50	2.51
G4	$5.99 \cdot 10^{-3}$	$3.99 \cdot 10^{-3}$	$2.39 \cdot 10^{-3}$	1.50	2.51
C4	$5.99 \cdot 10^{-3}$	$3.99 \cdot 10^{-3}$	$2.39 \cdot 10^{-3}$	1.50	2.51
T4	$5.99 \cdot 10^{-3}$	$3.99 \cdot 10^{-3}$	$2.39 \cdot 10^{-3}$	1.50	2.51
L5	$4.19 \cdot 10^{-3}$	$2.79 \cdot 10^{-3}$	$1.68 \cdot 10^{-3}$	1.50	2.49
B5	$4.19 \cdot 10^{-3}$	$2.79 \cdot 10^{-3}$	$1.68 \cdot 10^{-3}$	1.50	2.49
G5	$4.19 \cdot 10^{-3}$	$2.79 \cdot 10^{-3}$	$1.68 \cdot 10^{-3}$	1.50	2.49
C5	$4.19 \cdot 10^{-3}$	$2.79 \cdot 10^{-3}$	$1.68 \cdot 10^{-3}$	1.50	2.49
T5	$4.19 \cdot 10^{-3}$	$2.79 \cdot 10^{-3}$	$1.68 \cdot 10^{-3}$	1.50	2.49

TABLE 2.4.9 – Propagation du pulse gaussien.
Valeurs maximales de l'erreur L^2, schéma LF2.

Base	N=81 points	N=121 points	N=201 points	Gain 81 pts → 121 pts	Gain 81 pts → 201 pts
L1	$4.01 \cdot 10^{-2}$	$2.67 \cdot 10^{-2}$	$1.60 \cdot 10^{-2}$	1.50	2.51
B1	$4.02 \cdot 10^{-2}$	$2.67 \cdot 10^{-2}$	$1.60 \cdot 10^{-2}$	1.50	2.51
G1	$4.02 \cdot 10^{-2}$	$2.67 \cdot 10^{-2}$	$1.60 \cdot 10^{-2}$	1.50	2.51
C1	$4.02 \cdot 10^{-2}$	$2.67 \cdot 10^{-2}$	$1.60 \cdot 10^{-2}$	1.50	2.51
T1	$4.02 \cdot 10^{-2}$	$2.67 \cdot 10^{-2}$	$1.60 \cdot 10^{-2}$	1.50	2.51
L2	$1.80 \cdot 10^{-2}$	$1.20 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	1.50	2.51
B2	$1.80 \cdot 10^{-2}$	$1.20 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	1.50	2.51
G2	$1.80 \cdot 10^{-2}$	$1.20 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	1.50	2.51
C2	$1.80 \cdot 10^{-2}$	$1.20 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	1.50	2.51
T2	$1.80 \cdot 10^{-2}$	$1.20 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	1.50	2.51
L3	$1.08 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	$4.31 \cdot 10^{-3}$	1.50	2.51
B3	$1.08 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	$4.31 \cdot 10^{-3}$	1.50	2.51
G3	$1.08 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	$4.31 \cdot 10^{-3}$	1.50	2.51
C3	$1.08 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	$4.31 \cdot 10^{-3}$	1.50	2.51
T3	$1.08 \cdot 10^{-2}$	$7.18 \cdot 10^{-3}$	$4.31 \cdot 10^{-3}$	1.50	2.51
L4	$7.18 \cdot 10^{-3}$	$4.79 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	1.50	2.50
B4	$7.18 \cdot 10^{-3}$	$4.79 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	1.50	2.50
G4	$7.18 \cdot 10^{-3}$	$4.79 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	1.50	2.50
C4	$7.18 \cdot 10^{-3}$	$4.79 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	1.50	2.50
T4	$7.18 \cdot 10^{-3}$	$4.79 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	1.50	2.50
L5	$4.79 \cdot 10^{-3}$	$3.19 \cdot 10^{-3}$	$1.92 \cdot 10^{-3}$	1.50	2.49
B5	$4.79 \cdot 10^{-3}$	$3.19 \cdot 10^{-3}$	$1.92 \cdot 10^{-3}$	1.50	2.49
G5	$4.79 \cdot 10^{-3}$	$3.19 \cdot 10^{-3}$	$1.92 \cdot 10^{-3}$	1.50	2.49
C5	$4.79 \cdot 10^{-3}$	$3.19 \cdot 10^{-3}$	$1.92 \cdot 10^{-3}$	1.50	2.49
T5	$4.79 \cdot 10^{-3}$	$3.19 \cdot 10^{-3}$	$1.92 \cdot 10^{-3}$	1.50	2.49

TABLE 2.4.10 – Propagation du pulse gaussien.
Valeurs maximales de l'erreur L^2, schéma LF4.

Base	N=81 points	N=121 points	N=201 points	Gain 81 pts → 121 pts	Gain 81 pts → 201 pts
L1	$1.74 \cdot 10^{-2}$	$1.16 \cdot 10^{-2}$	$6.92 \cdot 10^{-3}$	1.50	2.51
B1	$1.72 \cdot 10^{-2}$	$1.15 \cdot 10^{-2}$	$6.90 \cdot 10^{-3}$	1.50	2.49
G1	$1.72 \cdot 10^{-2}$	$1.15 \cdot 10^{-2}$	$6.90 \cdot 10^{-3}$	1.50	2.49
C1	$1.72 \cdot 10^{-2}$	$1.15 \cdot 10^{-2}$	$6.90 \cdot 10^{-3}$	1.50	2.49
T1	$1.72 \cdot 10^{-2}$	$1.15 \cdot 10^{-2}$	$6.90 \cdot 10^{-3}$	1.50	2.49
L2	$1.06 \cdot 10^{-4}$	$3.13 \cdot 10^{-5}$	$1.04 \cdot 10^{-5}$	3.39	10.19
B2	$1.675 \cdot 10^{-5}$	$9.445 \cdot 10^{-6}$	$9.46 \cdot 10^{-6}$	1.77	1.77
G2	$1.675 \cdot 10^{-5}$	$9.445 \cdot 10^{-6}$	$9.46 \cdot 10^{-6}$	1.77	1.77
C2	$1.675 \cdot 10^{-5}$	$9.445 \cdot 10^{-6}$	$9.46 \cdot 10^{-6}$	1.77	1.77
T2	$1.675 \cdot 10^{-5}$	$9.445 \cdot 10^{-6}$	$9.46 \cdot 10^{-6}$	1.77	1.77
L3	$1.34 \cdot 10^{-5}$	$9.75 \cdot 10^{-6}$	$9.73 \cdot 10^{-6}$	1.37	1.38
B3	$1.31 \cdot 10^{-5}$	$9.75 \cdot 10^{-6}$	$9.73 \cdot 10^{-6}$	1.34	1.35
G3	$1.31 \cdot 10^{-5}$	$9.75 \cdot 10^{-6}$	$9.73 \cdot 10^{-6}$	1.34	1.35
C3	$1.31 \cdot 10^{-5}$	$9.75 \cdot 10^{-6}$	$9.73 \cdot 10^{-6}$	1.34	1.35
T3	$1.31 \cdot 10^{-5}$	$9.75 \cdot 10^{-6}$	$9.73 \cdot 10^{-6}$	1.34	1.35
L4	$9.54 \cdot 10^{-6}$	$9.70 \cdot 10^{-6}$	$9.82 \cdot 10^{-6}$	0.98	0.97
B4	$9.54 \cdot 10^{-6}$	$9.70 \cdot 10^{-6}$	$9.82 \cdot 10^{-6}$	0.98	0.97
G4	$9.56 \cdot 10^{-6}$	$9.71 \cdot 10^{-6}$	$9.82 \cdot 10^{-6}$	0.98	0.97
C4	$9.54 \cdot 10^{-6}$	$9.70 \cdot 10^{-6}$	$9.82 \cdot 10^{-6}$	0.98	0.97
T4	$9.54 \cdot 10^{-6}$	$9.70 \cdot 10^{-6}$	$9.82 \cdot 10^{-6}$	0.98	0.97
L5	$9.66 \cdot 10^{-6}$	$9.78 \cdot 10^{-6}$	$9.87 \cdot 10^{-6}$	0.99	0.98
B5	$9.66 \cdot 10^{-6}$	$9.78 \cdot 10^{-6}$	$9.87 \cdot 10^{-6}$	0.99	0.98
G5	$9.66 \cdot 10^{-6}$	$9.78 \cdot 10^{-6}$	$9.87 \cdot 10^{-6}$	0.99	0.98
C5	$9.66 \cdot 10^{-6}$	$9.78 \cdot 10^{-6}$	$9.87 \cdot 10^{-6}$	0.99	0.98
T5	$9.32 \cdot 10^{-6}$	$9.54 \cdot 10^{-6}$	$9.73 \cdot 10^{-6}$	0.98	0.96

TABLE 2.4.11 – Propagation du pulse gaussien.
Valeurs maximales de l'erreur L^2, schéma RK4.

Base	Gain LF2 → LF4	Gain LF2 → RK4	Gain LF4 → RK4
L1	0.82	1.89	2.30
B1	0.82	1.92	2.34
G1	0.82	1.92	2.34
C1	0.82	1.92	2.34
T1	0.82	1.92	2.34
L2	0.80	135.85	169.81
B2	0.80	859.70	1074.63
G2	0.80	859.70	1074.63
C2	0.80	859.70	1074.63
T2	0.80	859.70	1074.63
L3	0.83	670.15	805.97
B3	0.83	685.50	824.43
G3	0.83	685.50	824.43
C3	0.83	685.50	824.43
T3	0.83	685.50	824.43
L4	0.83	627.88	752.62
B4	0.83	627.88	752.62
G4	0.83	626.57	751.05
C4	0.83	627.88	752.62
T4	0.83	627.88	752.62
L5	0.87	433.75	495.86
B5	0.87	433.75	495.86
G5	0.87	433.75	495.86
C5	0.87	433.75	495.86
T5	0.87	449.57	513.95

TABLE 2.4.12 – Propagation du pulse gaussien.
Comparatif des gains obtenus sur l'erreur maximale L^2
entre les différents schémas en temps pour N=81 points.

L'évolution de l'erreur L^2 pour chacune des bases et pour chacun des schémas en temps est donnée sur les figures 2.28 à 2.32 pour le maillage à 81 points. Il en ressort que le schéma d'intégration en temps RK4 semble le plus précis mais également le plus instable pour la propagation d'un pulse triangulaire. L'étude des tableaux 2.4.13 à 2.4.15 indiquent que le schéma RK4 donne les meilleurs résultats de précision et que les schémas de type saute mouton ont des valeurs d'erreur très proches d'un tableau à l'autre bien qu'il semble que le schéma en temps LF2 réagisse mieux au h-raffinement que le schéma LF4. On note que désormais, certains résultats diffèrent d'une base à l'autre, ce qui laisse penser qu'un signal moins lisse devrait permettre de juger de l'intérêt d'une base plutôt qu'une autre.

Les derniers tableaux 2.4.16 à 2.4.18 nous permettent alors de vérifier que le schéma RK4 est bien le plus avantageux à utiliser pour ce cas test, bien que la différence avec les autres schémas ne soit pas aussi importante que dans le cas du pulse gaussien. On remarque par ailleurs que cette fois ci, le gain le plus important ne concerne plus l'interpolation quadratique comme ce fut le cas pour la propagation du pulse gaussien mais une interpolation d'ordre 4.

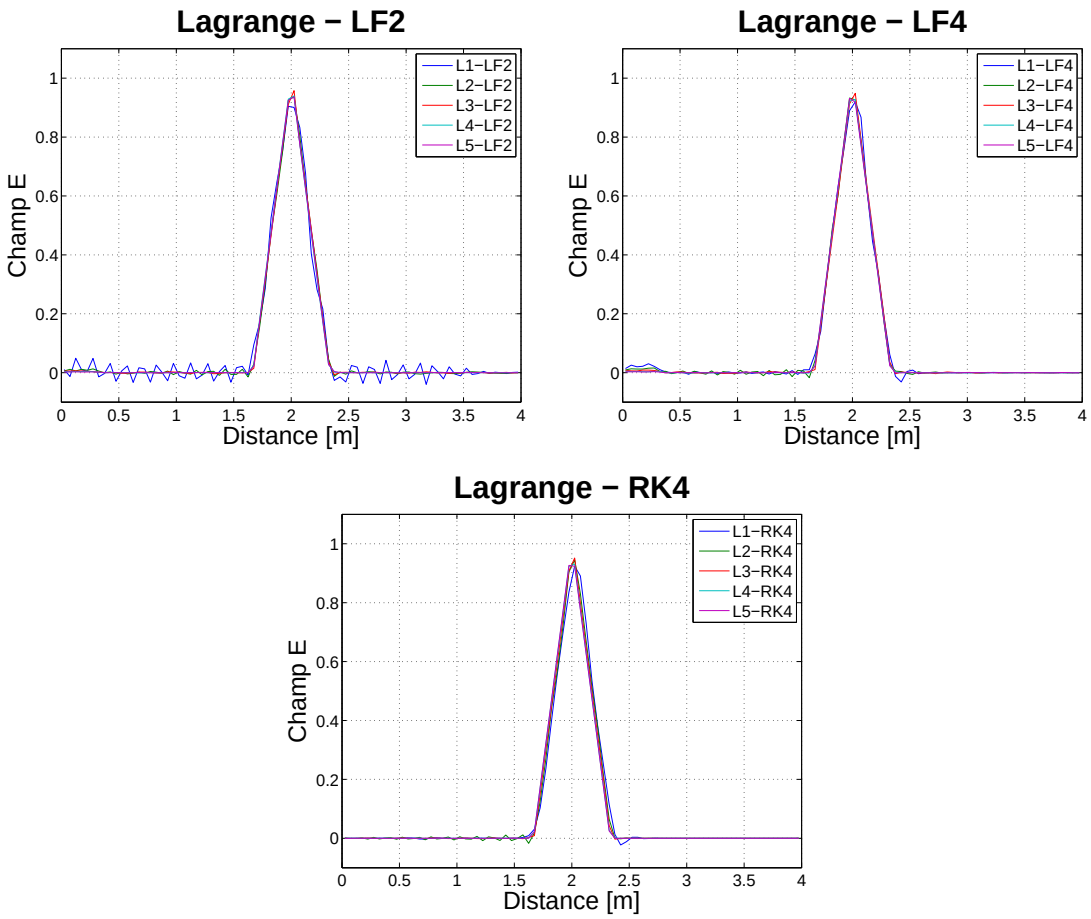


Figure 2.25 – Propagation du pulse triangulaire.
Champ électrique à $t = T_f$, N=81 points.

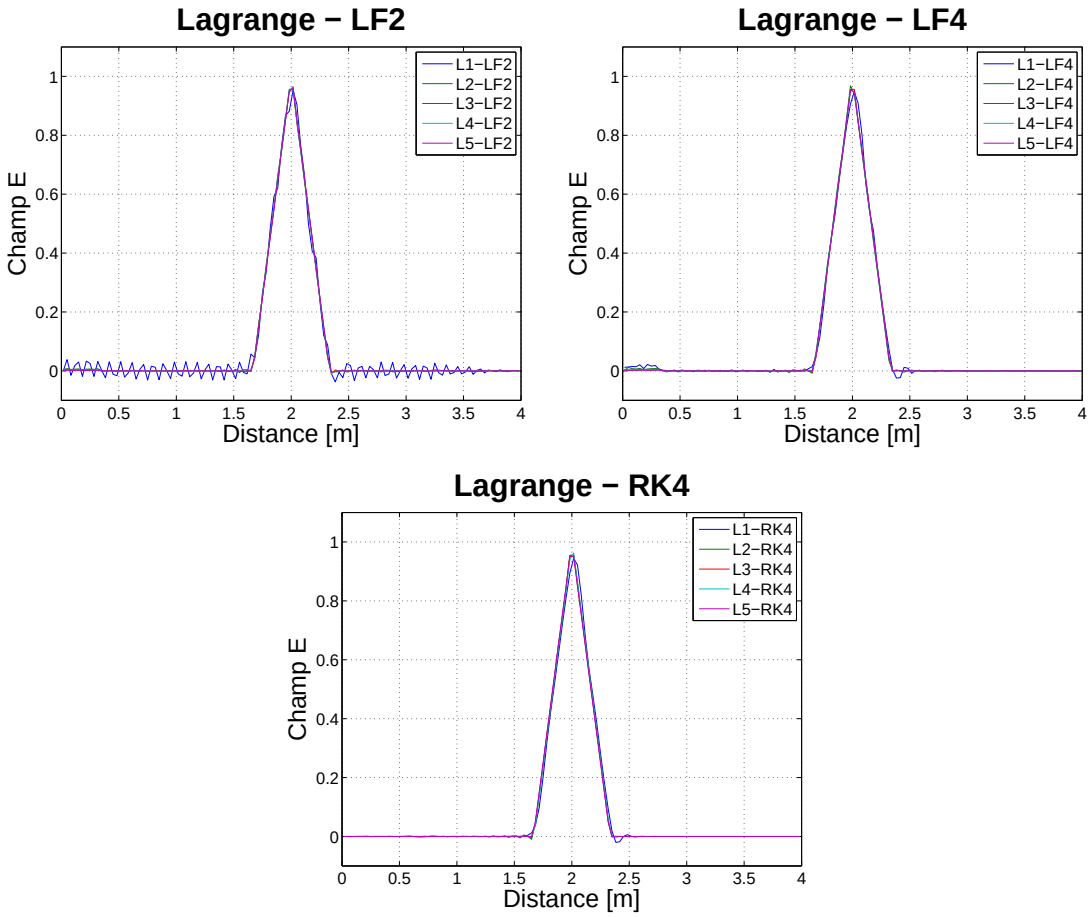


Figure 2.26 – Propagation du pulse triangulaire.
Champ électrique à $t = T_f$, N=121 points.

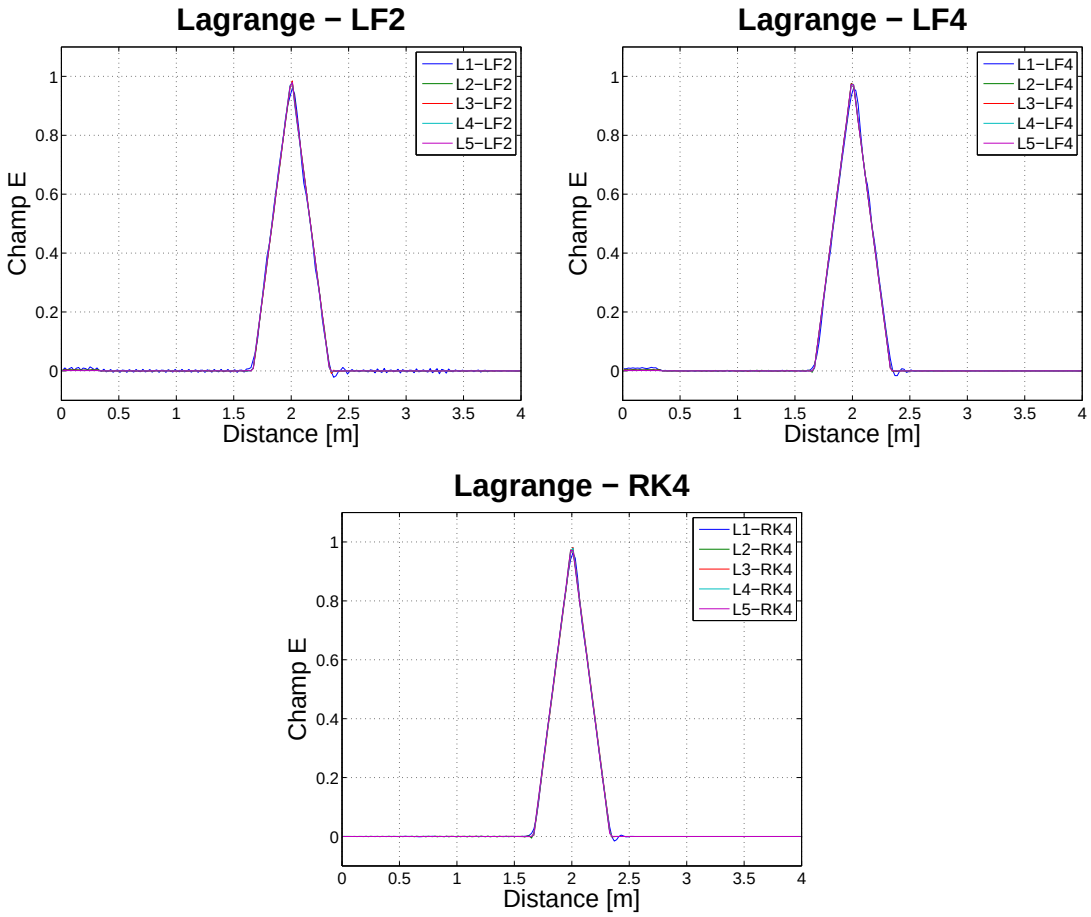


Figure 2.27 – Propagation du pulse triangulaire.
Champ électrique à $t = T_f$, N=201 points.

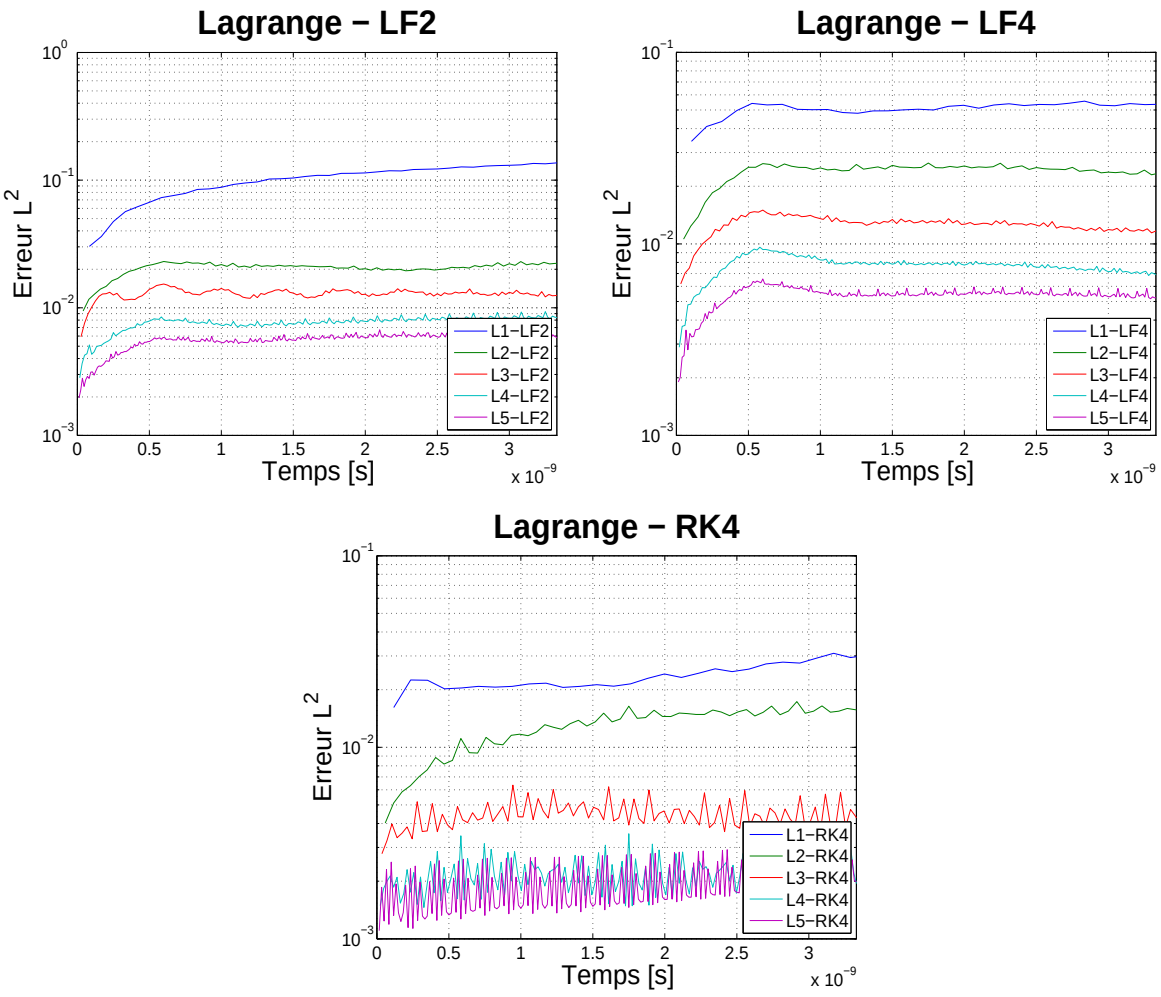


Figure 2.28 – Propagation du pulse triangulaire.
Evolution temporelle de l'erreur L^2, N=81 points.

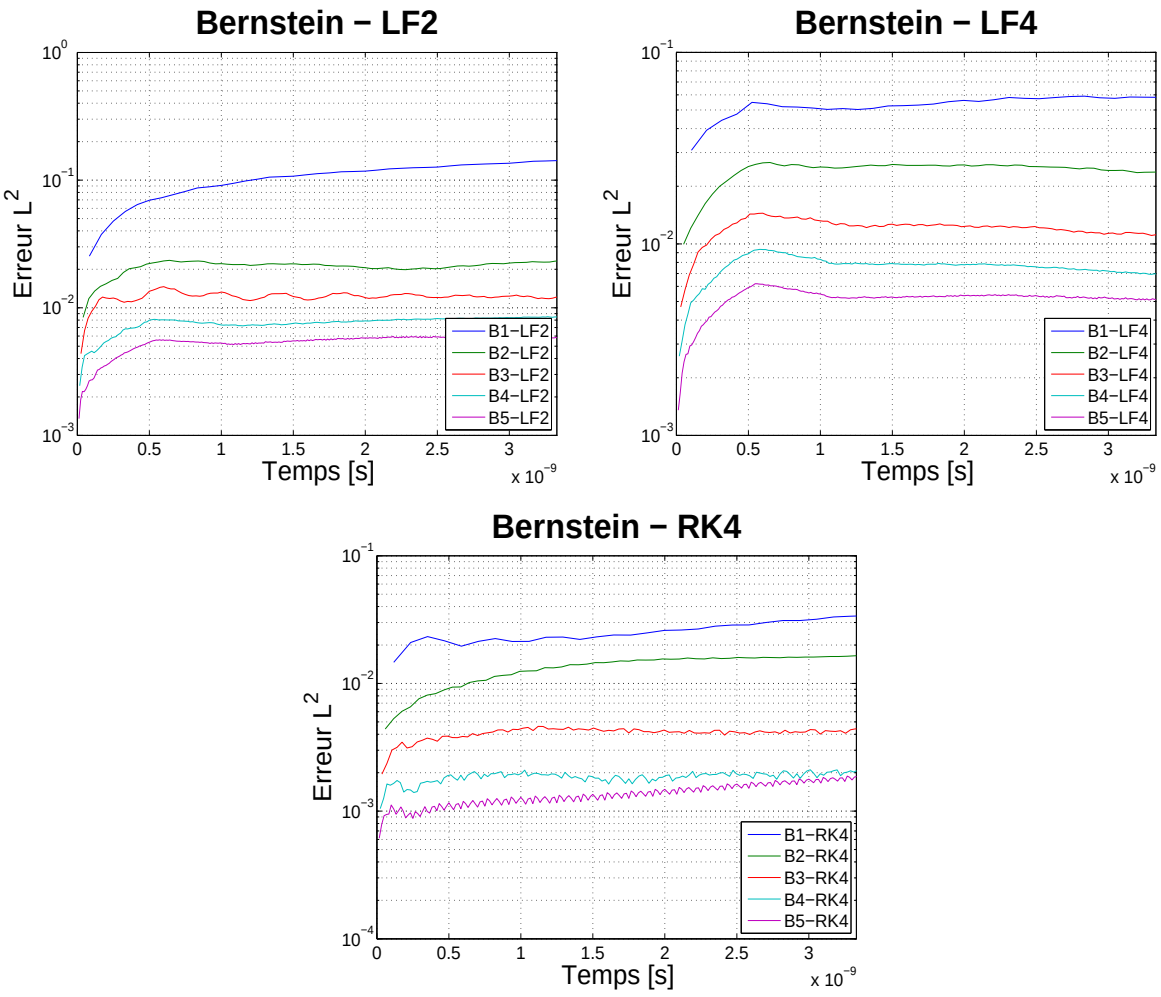


Figure 2.29 – Propagation du pulse triangulaire.
Evolution temporelle de l'erreur L^2, N=81 points.

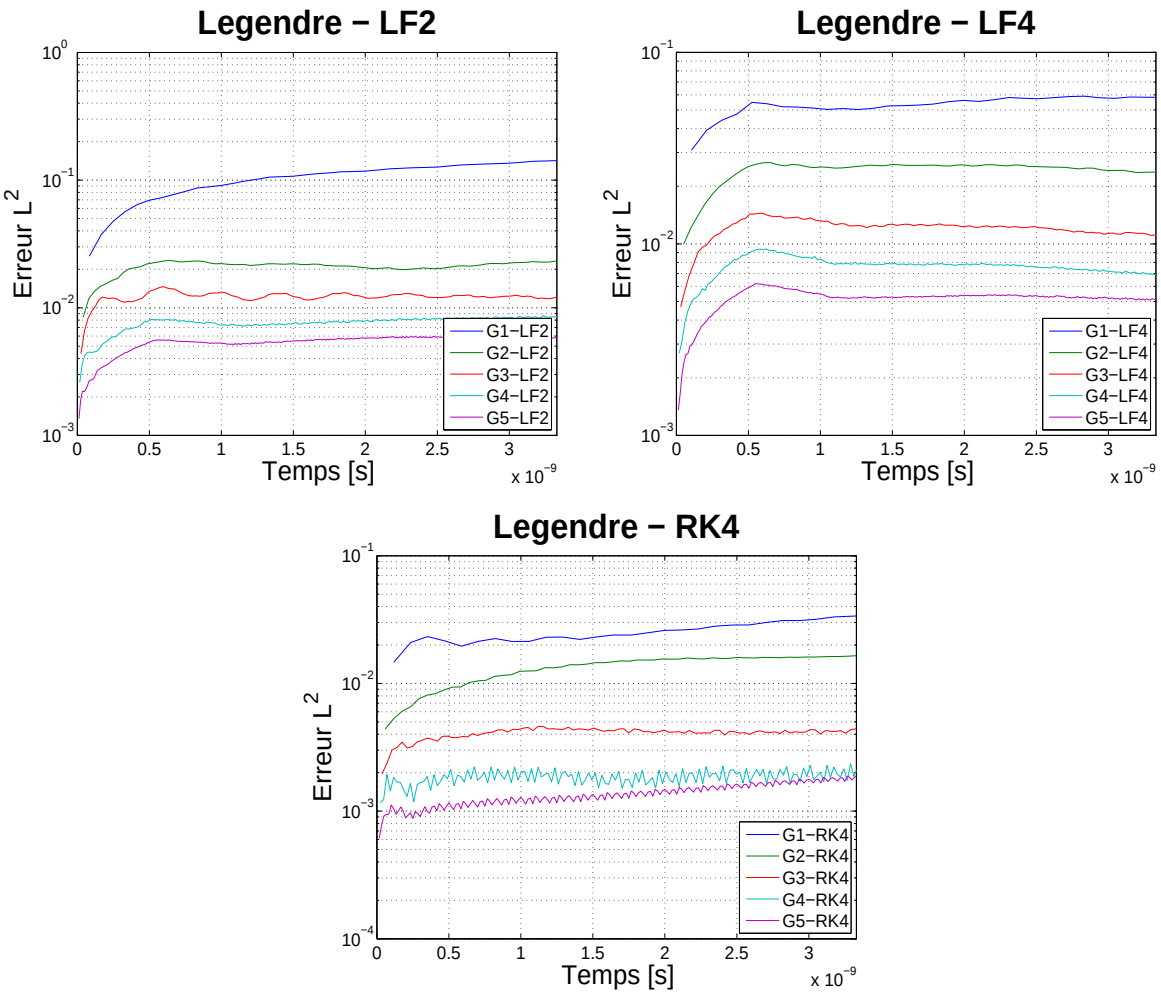


Figure 2.30 – Propagation du pulse triangulaire.
Evolution temporelle de l'erreur L^2, N=81 points.

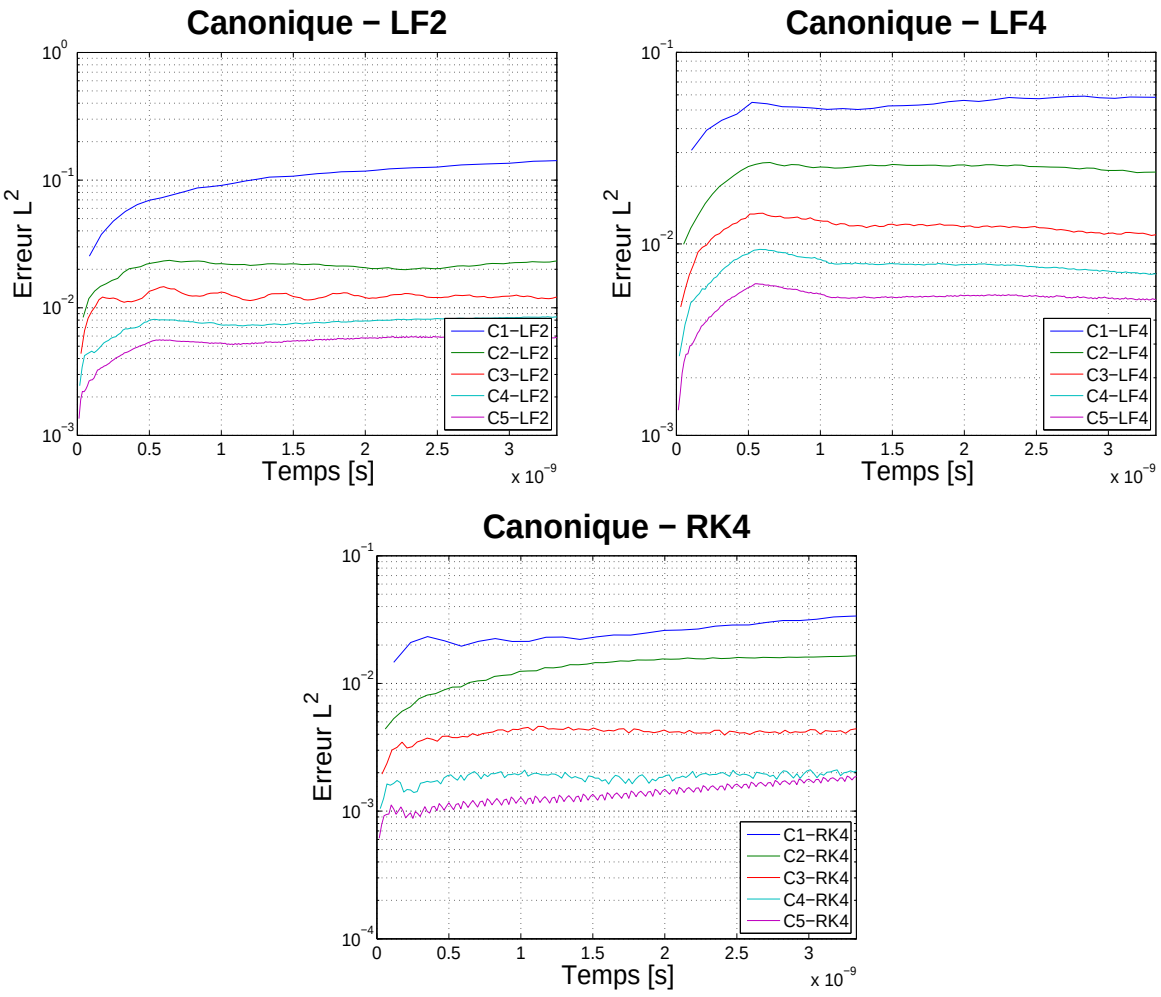


Figure 2.31 – Propagation du pulse triangulaire.
Evolution temporelle de l'erreur L^2, N=81 points.

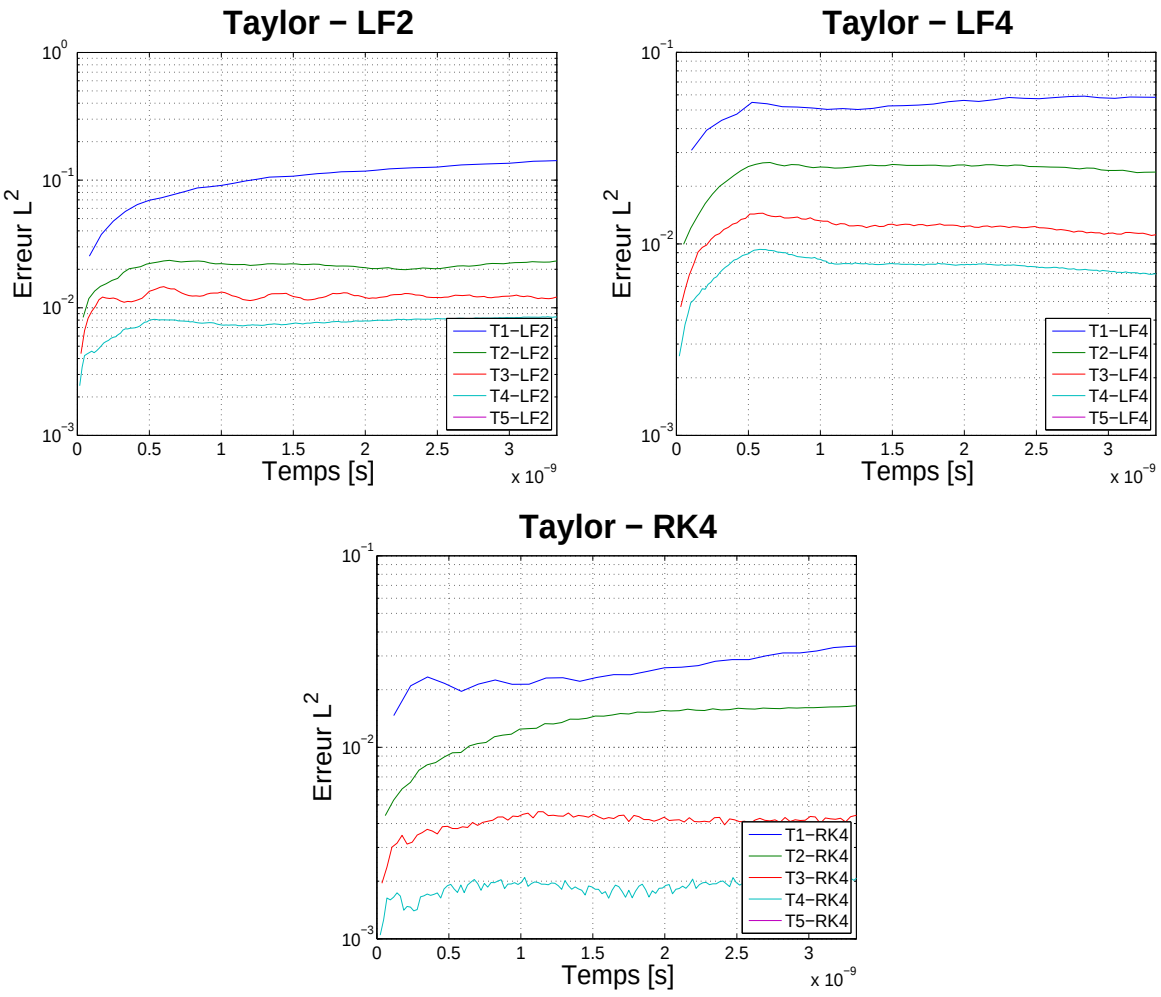


Figure 2.32 – Propagation du pulse triangulaire.
Evolution temporelle de l'erreur L^2, N=81 points.

Base	N=81 points	N=121 points	N=201 points	Gain 81 pts → 121 pts	Gain 81 pts → 201 pts
L1	$1.37 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$	$3.15 \cdot 10^{-2}$	1.11	4.35
B1	$1.42 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$	$3.41 \cdot 10^{-2}$	1.15	4.16
G1	$1.42 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$	$3.41 \cdot 10^{-2}$	1.15	4.16
C1	$1.42 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$	$3.41 \cdot 10^{-2}$	1.15	4.16
T1	$1.42 \cdot 10^{-1}$	$1.23 \cdot 10^{-1}$	$3.41 \cdot 10^{-2}$	1.15	4.16
L2	$2.30 \cdot 10^{-2}$	$1.60 \cdot 10^{-2}$	$8.30 \cdot 10^{-3}$	1.44	2.77
B2	$2.35 \cdot 10^{-2}$	$1.59 \cdot 10^{-2}$	$8.40 \cdot 10^{-3}$	1.48	2.80
G2	$2.35 \cdot 10^{-2}$	$1.59 \cdot 10^{-3}$	$8.40 \cdot 10^{-3}$	1.48	2.80
C2	$2.35 \cdot 10^{-2}$	$1.59 \cdot 10^{-3}$	$8.40 \cdot 10^{-3}$	1.48	2.80
T2	$2.35 \cdot 10^{-2}$	$1.59 \cdot 10^{-3}$	$8.40 \cdot 10^{-3}$	1.48	2.80
L3	$1.53 \cdot 10^{-2}$	$8.75 \cdot 10^{-3}$	$5.32 \cdot 10^{-3}$	1.75	2.88
B3	$1.46 \cdot 10^{-2}$	$8.64 \cdot 10^{-3}$	$4.96 \cdot 10^{-3}$	1.69	2.94
G3	$1.46 \cdot 10^{-2}$	$8.64 \cdot 10^{-3}$	$4.96 \cdot 10^{-3}$	1.69	2.94
C3	$1.46 \cdot 10^{-2}$	$8.64 \cdot 10^{-3}$	$4.96 \cdot 10^{-3}$	1.69	2.94
T3	$1.46 \cdot 10^{-2}$	$8.64 \cdot 10^{-3}$	$4.96 \cdot 10^{-4}$	1.69	2.94
L4	$9.33 \cdot 10^{-3}$	$6.42 \cdot 10^{-3}$	$3.43 \cdot 10^{-3}$	1.45	2.72
B4	$8.51 \cdot 10^{-3}$	$5.76 \cdot 10^{-3}$	$3.27 \cdot 10^{-3}$	1.48	2.60
G4	$8.62 \cdot 10^{-3}$	$5.85 \cdot 10^{-3}$	$3.29 \cdot 10^{-3}$	1.47	2.62
C4	$8.51 \cdot 10^{-3}$	$5.76 \cdot 10^{-3}$	$3.27 \cdot 10^{-3}$	1.48	2.60
T4	$8.51 \cdot 10^{-3}$	$5.76 \cdot 10^{-3}$	$3.27 \cdot 10^{-4}$	1.48	2.60
L5	$6.72 \cdot 10^{-3}$	$4.49 \cdot 10^{-3}$	$2.40 \cdot 10^{-3}$	1.50	2.80
B5	$5.97 \cdot 10^{-3}$	$3.97 \cdot 10^{-3}$	$2.26 \cdot 10^{-3}$	1.50	2.64
G5	$5.97 \cdot 10^{-3}$	$3.97 \cdot 10^{-3}$	$2.26 \cdot 10^{-3}$	1.50	2.64
C5	$5.97 \cdot 10^{-3}$	$3.97 \cdot 10^{-3}$	$2.26 \cdot 10^{-3}$	1.50	2.64

TABLE 2.4.13 – Propagation du pulse triangulaire.
Valeurs maximales de l'erreur L^2, schéma LF2.

Base	N=81 points	N=121 points	N=201 points	Gain 81 pts → 121 pts	Gain 81 pts → 201 pts
L1	$5.565 \cdot 10^{-2}$	$3.79 \cdot 10^{-2}$	$2.18 \cdot 10^{-2}$	1.47	2.55
B1	$5.91 \cdot 10^{-2}$	$3.82 \cdot 10^{-2}$	$2.27 \cdot 10^{-2}$	1.55	2.60
G1	$5.91 \cdot 10^{-2}$	$3.82 \cdot 10^{-2}$	$2.27 \cdot 10^{-2}$	1.55	2.60
C1	$5.91 \cdot 10^{-2}$	$3.82 \cdot 10^{-2}$	$2.27 \cdot 10^{-2}$	1.55	2.60
T1	$5.91 \cdot 10^{-2}$	$3.82 \cdot 10^{-2}$	$2.27 \cdot 10^{-2}$	1.55	2.60
L2	$2.64 \cdot 10^{-2}$	$1.77 \cdot 10^{-2}$	$1.01 \cdot 10^{-2}$	1.49	2.61
B2	$2.66 \cdot 10^{-2}$	$1.79 \cdot 10^{-2}$	$1.02 \cdot 10^{-2}$	1.49	2.61
G2	$2.66 \cdot 10^{-2}$	$1.79 \cdot 10^{-2}$	$1.02 \cdot 10^{-2}$	1.49	2.61
C2	$2.66 \cdot 10^{-2}$	$1.79 \cdot 10^{-2}$	$1.02 \cdot 10^{-2}$	1.49	2.61
T2	$2.66 \cdot 10^{-2}$	$1.79 \cdot 10^{-2}$	$1.02 \cdot 10^{-2}$	1.49	2.61
L3	$1.50 \cdot 10^{-2}$	$9.60 \cdot 10^{-3}$	$5.68 \cdot 10^{-3}$	1.56	2.64
B3	$1.45 \cdot 10^{-2}$	$9.64 \cdot 10^{-3}$	$5.58 \cdot 10^{-3}$	1.50	2.60
G3	$1.45 \cdot 10^{-2}$	$9.64 \cdot 10^{-3}$	$5.58 \cdot 10^{-3}$	1.50	2.60
C3	$1.45 \cdot 10^{-2}$	$9.64 \cdot 10^{-3}$	$5.58 \cdot 10^{-3}$	1.50	2.60
T3	$1.45 \cdot 10^{-2}$	$9.64 \cdot 10^{-3}$	$5.58 \cdot 10^{-3}$	1.50	2.60
L4	$9.61 \cdot 10^{-3}$	$6.37 \cdot 10^{-3}$	$3.70 \cdot 10^{-3}$	1.51	2.60
B4	$9.36 \cdot 10^{-3}$	$6.20 \cdot 10^{-3}$	$3.67 \cdot 10^{-3}$	1.51	2.55
G4	$9.40 \cdot 10^{-3}$	$6.22 \cdot 10^{-3}$	$3.67 \cdot 10^{-3}$	1.51	2.56
C4	$9.36 \cdot 10^{-3}$	$6.20 \cdot 10^{-3}$	$3.67 \cdot 10^{-3}$	1.51	2.55
T4	$9.36 \cdot 10^{-3}$	$6.20 \cdot 10^{-3}$	$3.67 \cdot 10^{-3}$	1.51	2.55
L5	$6.56 \cdot 10^{-3}$	$4.32 \cdot 10^{-3}$	$2.52 \cdot 10^{-3}$	1.52	2.60
B5	$6.19 \cdot 10^{-3}$	$4.13 \cdot 10^{-3}$	$2.45 \cdot 10^{-3}$	1.50	2.53
G5	$6.19 \cdot 10^{-3}$	$4.13 \cdot 10^{-3}$	$2.45 \cdot 10^{-3}$	1.50	2.53
C5	$6.19 \cdot 10^{-3}$	$4.13 \cdot 10^{-3}$	$2.45 \cdot 10^{-3}$	1.50	2.53

TABLE 2.4.14 – Propagation du pulse triangulaire.
Valeurs maximales de l'erreur L^2, schéma LF4.

Base	N=81 points	N=121 points	N=201 points	Gain 81 pts → 121 pts	Gain 81 pts → 201 pts
L1	$3.09 \cdot 10^{-2}$	$2.21 \cdot 10^{-2}$	$1.45 \cdot 10^{-2}$	1.40	2.13
B1	$3.39 \cdot 10^{-2}$	$2.38 \cdot 10^{-2}$	$1.53 \cdot 10^{-2}$	1.42	2.22
G1	$3.39 \cdot 10^{-2}$	$2.38 \cdot 10^{-2}$	$1.53 \cdot 10^{-2}$	1.42	2.22
C1	$3.39 \cdot 10^{-2}$	$2.38 \cdot 10^{-2}$	$1.53 \cdot 10^{-2}$	1.42	2.22
T1	$3.39 \cdot 10^{-2}$	$2.38 \cdot 10^{-2}$	$1.53 \cdot 10^{-2}$	1.42	2.22
L2	$1.73 \cdot 10^{-2}$	$9.64 \cdot 10^{-3}$	$4.29 \cdot 10^{-3}$	1.79	4.03
B2	$1.65 \cdot 10^{-2}$	$8.77 \cdot 10^{-3}$	$4.07 \cdot 10^{-3}$	1.88	4.05
G2	$1.65 \cdot 10^{-2}$	$8.77 \cdot 10^{-3}$	$4.07 \cdot 10^{-3}$	1.88	4.05
C2	$1.65 \cdot 10^{-2}$	$8.77 \cdot 10^{-3}$	$4.07 \cdot 10^{-3}$	1.88	4.05
T2	$1.65 \cdot 10^{-2}$	$8.77 \cdot 10^{-3}$	$4.07 \cdot 10^{-3}$	1.88	4.05
L3	$6.34 \cdot 10^{-3}$	$3.99 \cdot 10^{-3}$	$1.82 \cdot 10^{-3}$	1.59	3.48
B3	$4.61 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	1.61	3.36
G3	$4.61 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	1.61	3.36
C3	$4.61 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	1.61	3.36
T3	$4.61 \cdot 10^{-3}$	$2.87 \cdot 10^{-3}$	$1.37 \cdot 10^{-3}$	1.61	3.36
L4	$3.71 \cdot 10^{-3}$	$2.51 \cdot 10^{-3}$	$1.10 \cdot 10^{-3}$	1.48	3.37
B4	$2.11 \cdot 10^{-3}$	$1.46 \cdot 10^{-3}$	$7.42 \cdot 10^{-4}$	1.45	2.84
G4	$2.34 \cdot 10^{-3}$	$1.55 \cdot 10^{-3}$	$7.93 \cdot 10^{-4}$	1.51	2.95
C4	$2.11 \cdot 10^{-3}$	$1.46 \cdot 10^{-3}$	$7.42 \cdot 10^{-4}$	1.45	2.84
T4	$2.11 \cdot 10^{-3}$	$1.46 \cdot 10^{-3}$	$7.42 \cdot 10^{-4}$	1.45	2.84
L5	$3.12 \cdot 10^{-3}$	$2.23 \cdot 10^{-3}$	$1.05 \cdot 10^{-3}$	1.40	2.97
B5	$1.88 \cdot 10^{-3}$	$1.35 \cdot 10^{-3}$	$8.05 \cdot 10^{-4}$	1.39	2.34
G5	$1.88 \cdot 10^{-3}$	$1.35 \cdot 10^{-3}$	$8.05 \cdot 10^{-4}$	1.39	2.34
C5	$1.88 \cdot 10^{-3}$	$1.35 \cdot 10^{-3}$	$8.05 \cdot 10^{-4}$	1.39	2.34

TABLE 2.4.15 – Propagation du pulse triangulaire.
Valeurs maximales de l'erreur L^2, schéma RK4.

Base	Gain LF2 → LF4	Gain LF2 → RK4	Gain LF4 → RK4
L1	2.46	4.43	1.80
B1	2.40	4.19	1.74
G1	2.40	4.19	1.74
C1	2.40	4.19	1.74
T1	2.40	4.19	1.74
L2	0.87	1.33	1.53
B2	0.88	1.42	1.61
G2	0.88	1.42	1.61
C2	0.88	1.42	1.61
T2	0.88	1.42	1.61
L3	1.02	2.41	2.37
B3	1.01	3.17	3.15
G3	1.01	3.17	3.15
C3	1.01	3.17	3.15
T3	1.01	3.17	3.15
L4	0.97	2.51	2.59
B4	0.91	4.03	4.44
G4	0.92	3.68	4.02
C4	0.91	4.03	4.44
T4	0.91	4.03	4.44
L5	1.02	2.15	2.10
B5	1.04	3.18	3.29
G5	1.04	3.18	3.29
C5	1.04	3.18	3.29
T5	1.04	3.18	3.29

TABLE 2.4.16 – Propagation du pulse triangulaire.
Comparatif des gains obtenus sur l'erreur maximale L^2
entre les différents schémas en temps pour N=81 points.

Base	Gain LF2 → LF4	Gain LF2 → RK4	Gain LF4 → RK4
L1	3.25	5.57	1.71
B1	3.22	5.17	1.61
G1	3.22	5.17	1.61
C1	3.22	5.17	1.61
T1	3.22	5.17	1.61
L2	0.90	1.66	1.84
B2	0.89	1.81	2.04
G2	0.89	1.81	2.04
C2	0.89	1.81	2.04
T2	0.89	1.81	2.04
L3	0.91	2.19	2.41
B3	0.90	3.01	3.36
G3	0.90	3.01	3.36
C3	0.90	3.01	3.36
T3	0.90	3.01	3.36
L4	1.01	2.56	2.54
B4	0.93	3.95	4.25
G4	0.94	3.77	4.01
C4	0.93	3.95	4.25
T4	0.93	3.95	4.25
L5	1.04	2.01	1.94
B5	0.96	2.94	3.06
G5	0.96	2.94	3.06
C5	0.96	2.94	3.06
T5	0.96	2.94	3.06

TABLE 2.4.17 – Propagation du pulse triangulaire.
Comparatif des gains obtenus sur l'erreur maximale L^2
entre les différents schémas en temps pour N=121 points.

Base	Gain LF2 → LF4	Gain LF2 → RK4	Gain LF4 → RK4
L1	1.44	2.17	1.50
B1	1.50	2.23	1.48
G1	1.50	2.23	1.48
C1	1.50	2.23	1.48
T1	1.50	2.23	1.48
L2	0.82	1.93	2.35
B2	0.82	2.06	2.51
G2	0.82	2.06	2.51
C2	0.82	2.06	2.51
T2	0.82	2.06	2.51
L3	0.94	2.92	3.12
B3	0.89	3.62	4.07
G3	0.89	3.62	4.07
C3	0.89	3.62	4.07
T3	0.89	3.62	4.07
L4	0.93	3.12	3.36
B4	1.12	4.41	4.95
G4	0.90	4.15	4.63
C4	1.12	4.41	4.95
T4	1.12	4.41	4.95
L5	0.95	2.29	2.40
B5	0.92	2.81	3.04
G5	0.92	2.81	3.04
C5	0.92	2.81	3.04
T5	0.92	2.81	3.04

TABLE 2.4.18 – Propagation du pulse triangulaire.
Comparatif des gains obtenus sur l'erreur maximale L^2
entre les différents schémas en temps pour N=201 points.

2.5 Bases hiérarchiques en 2D

2.5.1 Préambule

L'utilisation de simplexes rend la construction d'un jeu optimal de fonctions de base difficile. L'élaboration d'un développement modal élémentaire de type p se fonde sur une distribution locale non uniforme de l'ordre de l'approximation polynomiale, c'est à dire que les éléments adjacents dans le maillage portent différents ordres d'approximation polynomiale. Une condition nécessaire pour l'implémentation de cette caractéristique, qui est essentielle pour une stratégie *hp*-adaptative, est la séparation des degrés de liberté en contributions internes (*i.e.* associés avec l'intérieur de l'élément) et externes (*i.e.* associés avec la frontière de l'élément) regroupés au sein d'une même structure hiérarchique. Les modes frontières sont des modes ayant un support non nul sur la frontière de la région standard alors que les modes intérieurs correspondent aux modes s'annulant sur toutes les frontières. Dans la suite, nous désignons par modes de type *sommet* tous les modes qui valent un sur un sommet et qui s'annulent sur les autres sommets ; les modes de type *arête* par tous les modes ayant un support le long d'une arête et qui s'annulent sur toutes les autres arêtes et sommets ; et les modes de type *face* par tous les modes ayant un support sur une face et qui s'annulent sur toutes les autres faces, arêtes et sommets. Pour une approximation bidimensionnelle, les modes frontières sont constitués de modes de type arête et sommet alors qu'en dimension 3 les modes frontières contiennent les modes de type face, arête et sommet.

L'accent est porté sur l'obtention d'une famille optimale de fonctions générant l'espace des polynômes d'ordre au plus p_i sur l'élément τ_i. Les critères d'optimisation de la famille de polynômes sont alors :

- l'efficacité du calcul des intégrales de surface (pour les éléments de $\mathbb{P}_{p_i}$, seulement (p_i+1) fonctions en 2D et $(p_i+1)(p_i+2)/2$ en 3D devraient être non nulles sur une face) ;
- les intégrales de volume et de surface devraient être exactement connues si possible ;
- le conditionnement de la matrice de masse devrait être minimal ou l'inverse de la matrice de masse devrait être connue. Une approximation précise est typiquement un jeu de fonctions orthogonales ou presque orthogonales puisque l'orthogonalité conduisent à des matrices bien conditionnées.

Dans ce qui suit, on se place dans le cas d'un maillage affine composé de simplexes non dégénérés et on présente plusieurs jeux de fonctions de base hiérarchiques proposés par différents auteurs, possédant des propriétés intéressantes et qui répondent plus ou moins bien aux critères énoncés précédemment.

On considère un domaine K et un espace $\mathcal{P}$ de polynômes d'ordre au plus p de dimension n_p. Soit $B^p = (\varphi_1, \varphi_2, \ldots, \varphi_{n_p})$ une base dans l'espace $\mathcal{P}$ possédant la propriété suivante :

$$B^p \subset B^{p+1}.$$

Une telle base est appelée base *hiérarchique* puisque les fonctions de base correspondant à une approximation polynomiale d'ordre p constituent un sous-ensemble du jeu de fonctions de base correspondant à l'approximation polynomiale d'ordre $(p+1)$. Les éléments hiérarchiques permettent une distribution locale non uniforme de l'ordre de l'approximation polynomiale plus facilement que les éléments nodaux, les rendant ainsi appropriés à la mise en œuvre d'une stratégie *p*- et *hp*-adaptative.

2.5.2 Fonctions de base de Lobatto et fonctions noyau

Parmi les fonctions de base hiérarchiques, un des choix les plus populaires se porte sur les fonctions de base de Lobatto (polynômes de Legendre intégrés). Leurs excellentes propriétés de conditionnement proviennent du fait que leurs dérivées correspondent aux polynômes de Legendre (normalisés), et par conséquent que :

$$\int_{-1}^{1} l'_{i-1}(\xi) l'_{j-1}(\xi)\, d\xi = 0 \quad \text{whenever } i > 2 \text{ or } j > 2, \quad i \neq j. \tag{2.5.1}$$

Définissons les fonctions :

$$\begin{cases} l_0(x) &= \dfrac{1-x}{2}, \\ l_1(x) &= \dfrac{x+1}{2}, \\ l_k(x) &= \dfrac{1}{\|L_{k-1}\|_2}\displaystyle\int_{-1}^{x} L_{k-1}(\xi)\,d\xi, \quad 2 \le k, \end{cases}$$

où $\|L_{k-1}\|_2 = \sqrt{2/(2k-1)}$. De manière évidente $l_k(-1) = 0,\ k \ge 2$. Il suit de l'orthogonalité des polynômes d'ordre élevé de Legendre L_k à $L_0 \equiv 1$ que :

$$\int_{-1}^{1} L_k(x)dx = 0, \quad k \ge 1, \tag{2.5.2}$$

et aussi que $l_k(1) = 0$, $k \ge 2$. Les *fonctions de base de Lobatto* $l_0, l_1, \ldots, l_p$ forment une base complète de l'espace $P_p(-1,1)$ des polynômes d'ordre au plus p dans l'intervalle $[-1,1]$.

Pour la suite, il est préférable de décomposer les fonctions de base d'ordre élevé de Lobatto $l_2, l_3, \ldots$ en produits de la forme :

$$l_k(x) = l_0(x)\, l_1(x)\, \phi_{k-2}(x), \quad k \ge 2. \tag{2.5.3}$$

Puisque toutes les fonctions $l_k,\ k \ge 2$ disparaissent à ± 1, les *fonctions noyau* $\phi_{k-2},\ k \ge 2$ sont des *polynômes* d'ordre $k-2$.

2.5.3 Les polynômes de Solin

Dans leur ouvrage [Solin 2004], Solin *et al.* décrivent plusieurs familles de polynômes hiérarchiques correspondant à différents espaces d'approximation de régularité globale H^1, $H(\mathrm{rot})$ et $H(\mathrm{div})$. On présente ici la famille de polynômes correspondant à l'approximation de régularité globale H^1 en triangle. Soit :

$$K_t = \{\xi \in \mathbb{R}^2;\ 0 < \xi_1, \xi_2;\ \xi_1 + \xi_2 < 1\}, \tag{2.5.4}$$

l'élément triangulaire de référence d'ordre arbitraire représenté sur la figure 2.33.

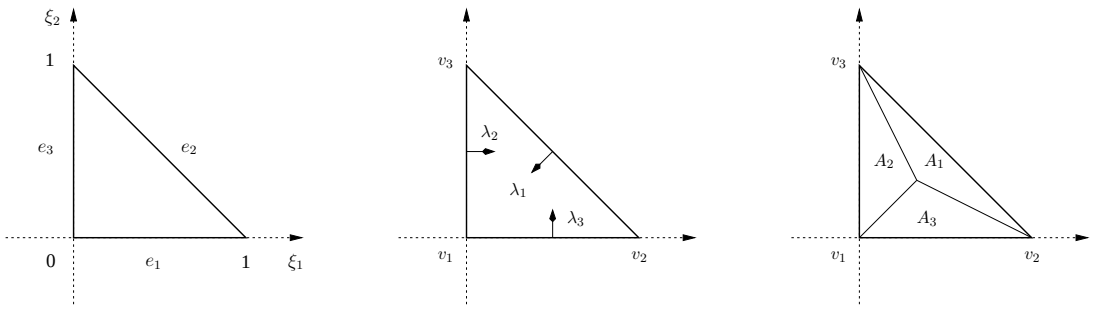


Figure 2.33 – Le triangle de référence et ses coordonnées barycentriques

La représentation élémentaire des fonctions de forme locales est fondée sur l'utilisation des systèmes de coordonnées barycentriques. Ce système de coordonnées est illustré sur la figure 2.33 pour le triangle standard. Chaque point du triangle est décrit par ses 3 coordonnées λ_1, λ_2 et λ_3 qui peuvent être interprétées comme le quotient des aires A_1, A_2 et A_3 sur l'aire totale $A = A_1 + A_2 + A_3$, *i.e.* :

$$\lambda_1 = \frac{A_1}{A}, \quad \lambda_2 = \frac{A_2}{A}, \quad \lambda_3 = \frac{A_3}{A}. \tag{2.5.5}$$

Par conséquent, λ_1, λ_2 et λ_3 ont une valeur unitaire sur les coins v_1, v_2 et v_3 de la figure 2.33, respectivement. Par définition, ces coordonnées vérifient la relation :

$$\lambda_1 + \lambda_2 + \lambda_3 = 1, \tag{2.5.6}$$

et peuvent être exprimées en terme de coordonnées cartésiennes ξ_1, ξ_2 comme suit :

$$\lambda_1(\xi_1, \xi_2) = 1 - \xi_1 - \xi_2, \quad \lambda_2(\xi_1, \xi_2) = \xi_1, \quad \lambda_3(\xi_1, \xi_2) = \xi_2. \tag{2.5.7}$$

Pour la définition des fonctions de base, on considère un ordre local d'approximation p_i^b à l'intérieur de l'élément. Les arêtes e_1, e_2 et e_3 sur la figure 2.33 seront assignées des ordres polynomiaux locaux $p_i^{e_1}, \ldots, p_i^{e_3}$. On définit tout d'abord les *fonctions sommets* $\varphi_t^{v_1}, \ldots, \varphi_t^{v_3}$. Elles sont assignées aux sommets $v_1, \ldots, v_3$: les $\varphi_t^{v_j}$ vallant l'unité sur le sommet v_j et s'annulant sur les deux sommets restants. Ces fonctions sont choisies linéaires en utilisant les coordonnées affines :

$$\left\{ \begin{array}{lcl} \varphi_t^{v_1}(\xi_1, \xi_2) & = & \lambda_1, \\ \varphi_t^{v_2}(\xi_1, \xi_2) & = & \lambda_2, \\ \varphi_t^{v_3}(\xi_1, \xi_2) & = & \lambda_3. \end{array} \right.$$

On définit ensuite les *fonctions arêtes* $\varphi_{k,t}^{e_j}$, $k = 2, \ldots, p_i^{e_j}$, $j = 1, \ldots, 3$. Les traces des fonctions arêtes $\varphi_{k,t}^{e_j}$ $k = 2, \ldots, p_i^{e_j}$ coïncident avec les fonctions de base de Lobatto $l_2, l_3, \ldots$ sur l'arête e_j, et s'annulent sur toutes les arêtes restantes. Nous pouvons les écrire sous la forme suivante :

$$\left\{ \begin{array}{lcll} \varphi_{k,t}^{e_1} & = & \lambda_1\lambda_2\, \phi_{k-2}(\lambda_2 - \lambda_1), & 2 \le k \le p_i^{e_1}, \\ \varphi_{k,t}^{e_2} & = & \lambda_2\lambda_3\, \phi_{k-2}(\lambda_3 - \lambda_2), & 2 \le k \le p_i^{e_2}, \\ \varphi_{k,t}^{e_3} & = & \lambda_3\lambda_1\, \phi_{k-2}(\lambda_3 - \lambda_1), & 2 \le k \le p_i^{e_3}. \end{array} \right.$$

La base hiérarchique est complétée en définissant des *fonctions de type bulle* qui s'annulent entièrement sur la frontière de l'élément. Ces fonctions sont internes, une approche standard est de simplement combiner les coordonnées affines avec des puissances variables :

$$\varphi_{n_1,n_2,t}^b = \lambda_1(\lambda_2)^{n_1}(\lambda_3)^{n_2}, \quad 1 \le n_1, n_2; \quad n_1 + n_2 \le p_i^b - 1. \tag{2.5.8}$$

Toutefois, à cause de leurs mauvaises propriétés de conditionnement, on définit un nouvel ensemble de fonctions de type bulle :

$$\varphi_{n_1,n_2,t}^b = \lambda_1\lambda_2\lambda_3\, \phi_{n_1-1}(\lambda_3 - \lambda_2)\, \phi_{n_2-1}(\lambda_2 - \lambda_1), \quad 1 \le n_1, n_2; \quad n_1 + n_2 \le p_i^b - 1. \tag{2.5.9}$$

Le tableau 2.5.19 quantifie le nombre de fonctions de forme hiérarchiques dans la base.

TABLE 2.5.19 – Fonctions de forme scalaires hiérarchiques

Entité géométrique	**Ordre polynomial**	**Nombre de fonctions de forme**
Sommet	toujours	1
Arête	$2 \le p_i^{e_j}$	$p_i^{e_j} - 1$
Intérieur	$3 \le p_i^b$	$(p_i^b - 1)(p_i^b - 2)/2$

2.5.4 Les polynômes d'Ainsworth-Coyle

Les polynômes introduits par Ainsworth et Coyle [Ainsworth 2003] sont basés sur les polynômes de Legendre et sont analogues de ceux de Solin et al. [Solin 2004] où les polynômes de Legendre sont remplaçés par des polynômes de Lobatto normalisés. Dans le triangle ces polynômes se répartissent en trois types qu'on indiquera avec un exposant s pour le type *sommet*, a le type *arête* et b pour le type *bulle*. Des polynômes de type *face* seront ajoutés dans le cas d'un tétraèdre. Ces polynômes s'écrivent en coordonnées barycentriques.

Définition 2.5.1 (Polynômes d'Ainsworth-Coyle degré k)

Modes de type sommet :

$$A_m^{s,1} = \lambda_m \quad m = 1, 2, 3.$$

Modes de type arête :

$$\begin{aligned} A^{\vec{a}_1,k} &= \lambda_1\lambda_2 P_{k-2}\left(\lambda_2 - \lambda_1\right), \\ A^{\vec{a}_2,k} &= \lambda_2\lambda_3 P_{k-2}\left(\lambda_3 - \lambda_2\right), \\ A^{\vec{a}_3,k} &= \lambda_3\lambda_1 P_{k-2}\left(\lambda_1 - \lambda_3\right). \end{aligned} \tag{2.5.10}$$

Modes de type bulle :

$$A_{m,n}^{b,k} = \lambda_1\lambda_2\lambda_3 P_{m-1}(\lambda_3 - \lambda_2) P_{n-1}(\lambda_2 - \lambda_1), \tag{2.5.11}$$

où dans (2.5.11) les paramètres k, m et n vérifient les conditions :

$$k \geq 3, \quad 1 \leq m, n, \quad (m+n) \leq (k-1),$$

et P_n est le polynôme de Legendre de degré n ie polynôme de Jacobi $P_n^{0,0}$.

La base de degré un étant identique à celle de Lagrange et de Bernstein nous donnons ici l'expression des polynômes pour les degrés deux et trois.

Exemple 2.5.1 *Base d'Ainsworth-Coyle de degré 2 dans un triangle*
On ajoute à la base de degré 1 trois polynômes de type arête :

$$\begin{aligned} A^{\vec{a}_1,2} &= \lambda_1\lambda_2, \\ A^{\vec{a}_2,2} &= \lambda_2\lambda_2, \\ A^{\vec{a}_3,2} &= \lambda_3\lambda_1. \end{aligned}$$

Exemple 2.5.2 *Base d'Ainsworth-Coyle de degré 3 dans un triangle*
On ajoute à la base de degré 2 trois polynômes de type arête et un polynôme de type bulle tous de degré trois :

$$\begin{aligned} A^{\vec{a}_1,3} &= \lambda_1\lambda_2(\lambda_2 - \lambda_1), \\ A^{\vec{a}_2,3} &= \lambda_2\lambda_3(\lambda_3 - \lambda_2), \\ A^{\vec{a}_3,3} &= \lambda_3\lambda_1(\lambda_1 - \lambda_3), \\ A_{1,1}^{b,3} &= \lambda_1\lambda_2\lambda_3. \end{aligned}$$

Les polynômes d'Ainsworth-Coyle ayant tous au moins une fonction barycentrique λ_m en facteur ils s'annulent sur l'arête correspondante (*i.e.* l'arête d'équation $\lambda_m = 0$). Avec la classification de ces polynômes en trois types on peut énoncer cette propriété de façon plus précise.

Proposition 2.5.1 *Tout polynôme de type arête $A^{a,k}_{mn}$ défini par (2.5.10) s'annule sur les arêtes d'équation $\lambda_m = 0$ et $\lambda_n = 0$.*

Proposition 2.5.2 *Tout polynôme de type bulle $A^{a,k}_{mn}$ défini dans un triangle s'annule sur le bord de ce triangle. C'est la conséquence directe d'avoir toutes les fonctions barycentriques en facteur (2.5.11) et pour cette raison on l'appelle aussi polynôme interne.*

2.5.5 Les polynômes de Sherwin-Karniadakis

Des bases modales et hiérarchiques dans le triangle et dans le tétraèdre ont été proposées par Sherwin et Karniadakis dans [Sherwin 1995]. Bien qu'on retrouve là aussi des polynômes de Jacobi ces bases ne sont pas une variante de celles d'Ainsworth-Coyle ou de Solin car ici le principe de base est l'utilisation de l'application singulière (*warped product*) qui transforme un rectangle en triangle (un cube en tétraédre en 3D) ce qui permet de revenir au produit tensoriel classique pour définir des polynômes de Jacobi en plusieurs variables (voir [Sherwin 1995] pour plus de détails). Notons que cette idée a été utilisée, probablement pour la première fois, par Dubiner [Dubiner 1991] pour construire ses bases hiérarchiques et orthogonales dans le triangle. On a préféré décrire ici les bases de Sherwin-Karniadakis car celles-ci s'expriment facilement en coordonnées barycentriques et surtout qu'elles vérifient la propriété (2.5.1) ce qui n'est pas le cas des bases de Dubiner.

Les polynômes de Sherwin-Karniadakis se répartissent aussi en trois types, *sommet*, *arête* et *bulle*, avec une différence notable avec les bases d'Ainsworth-Coyle ou de Solin car ici le type *arête* se décompose en trois cas bien distincts comme nous le voyons dans la définition de ces polynômes notés ici $K^{t,k}_{p,q}$.

Définition 2.5.2 (Polynômes de Sherwin-Karniadakis de degré k)

Modes de type sommet :

$$K^{s,1}_j = \lambda_j \quad j = 1, 2, 3.$$

Modes de type arête :

$$\begin{aligned} K^{\vec{a}_1,k}_{m,0} &= \lambda_1\lambda_2 P^{1,1}_{m-2}\left(\frac{\lambda_2-\lambda_1}{1-\lambda_3}\right)(1-\lambda_3)^{m-2}, \\ K^{\vec{a}_2,k}_{1,n} &= \lambda_2\lambda_3 P^{1,1}_{n-1}(2\lambda_3-1), \\ K^{\vec{a}_3,k}_{1,n} &= \lambda_3\lambda_1 P^{1,1}_{n-1}(2\lambda_3-1), \end{aligned} \tag{2.5.12}$$

où $k \geq 2, \quad k \geq m \geq 2, \quad k > n \geq 1$.
Modes de type bulle :

$$K^{b,k}_{m,n} = \lambda_1\lambda_2\lambda_3 P^{1,1}_{m-2}\left(\frac{\lambda_2-\lambda_1}{1-\lambda_3}\right)(1-\lambda_3)^{m-2} P^{2m-1,1}_{n-1}(2\lambda_3-1), \tag{2.5.13}$$

où $k \geq 3, \quad m \geq 2, \quad n \geq 1, \quad (m+n) \leq k$.
Ici $\vec{a}_j$ désigne la $j^{ième}$ arête du triangle orienté dans le sens trigonométrique et $P^{p,q}_n$ est le polynôme de Jacobi de degré n et de paramètres p, q qui sont ici des entiers. Pout être complet, on précise l'expression de $P^{p,q}_0$ et $P^{p,q}_1$:

$$P^{p,q}_0(x) = 1, \tag{2.5.14}$$

$$P^{p,q}_1(x) = \frac{p-q}{2} + (1+\frac{p+q}{2})x.$$

Remarque 2.5.1 *On peut s'inquiéter de voir le terme* $(1-\lambda_3)$ *en dénominateur dans l'expression des polynômes, cause certaine d'une singularité car ce terme s'annule précisément sur le troisième sommet du triangle* ($\lambda_3(S_3)=1$)*. Ce terme provient de l'application singulière qui transforme le carré en triangle [Sherwin 1995]. On notera cependant qu'on a aussi en facteur le terme* $(1-\lambda_3)^{m-2}$ *dont le rôle est justement d'éliminer cette singularité.*

*Nous avons retranscrit l'expression des polynômes dans le cas général (*i.e. *degré variable par élément) mais dans le cas uniforme supposé ici les paramètres* m *et* n *dans (2.5.12) se réduisent à* k *et* $k-1$ *respectivement.*

Par définition la base de degré un est identique à celle d'Ainsworth-Coyle, de Bernstein et de Lagrange de même degré. On obtient la base de degré deux en posant $k=2$, $m=2$, $n=1$ dans (2.5.12) et en utilisant (2.5.14) on retrouve la base d'Ainsworth-Coyle et de Bernstein. C'est à partir du degré trois que ces bases se différencient nettement et nous donnons ci-dessous l'expression des polynômes $K_{p,q}^{t,3}$.

Exemple 2.5.3 *Base de Sherwin-Karniadakis de degré 3 dans un triangle*

On ajoute à la base de degré deux (2.5.1) trois polynômes de type arête et un polynôme de type bulle tous de degré trois :

$$\begin{aligned} K_{3,0}^{\vec{a}_1,3} &= 2\lambda_1\lambda_2(\lambda_2-\lambda_1), \\ K_{1,2}^{\vec{a}_2,3} &= 2\lambda_2\lambda_3(2\lambda_3-1), \\ K_{1,2}^{\vec{a}_3,3} &= 2\lambda_3\lambda_1(2\lambda_3-1), \\ K_{2,1}^{b,3} &= \lambda_1\lambda_2\lambda_3. \end{aligned}$$

On déduit de la définition (2.5.2) que les propriétés (2.5.1) et (2.5.2) sont ici aussi vérifiées.

2.6 Etude numérique en 2D

2.6.1 Mode propre dans une cavité carrée parfaitement conductrice

Le problème test considéré est la propagation d'un mode propre dans une cavité carrée unitaire parfaitement conductrice. On utilise des maillages triangulaires uniformes obtenus par subdivision d'une grille cartésienne de N×N points. Chaque cellule de cette grille est divisée en deux triangles comme illustré sur la figure 2.34. Le nombre de triangles d'un tel maillage est donc égal à 2×(N-1)×(N-1). Le temps de simulation est fixé à T=3.3×10^{-8} s. La solution analytique des équations de Maxwell étant connue pour ce problème, on peut évaluer l'erreur L^2 entre les solutions analytique et numérique. Les comparaisons présentées ici se basent principalement sur cette quantité ainsi que sur les temps de calcul. On notera que les temps de calcul précisés dans les tableaux ci-dessous ne prennent pas en compte les opérations auxiliaires telles que le calcul de l'erreur L^2 ou la projection de la solution numérique en certains points particuliers en vue de suivre l'évolution temporelle des composantes du champ électromagnétique. Ces opérations sont particulièrement coûteuses pour les bases de fonctions modales pour lesquelles il faut faire appel à une procédure de projection adaptée à chaque évaluation de la solution analytique aux degrés de liberté de l'approximation.

Le tableau 2.6.20 regroupe les résultats de performance obtenus avec la formulation GDDT-$\mathbb{P}_p$ proposée par [Fezoui 2005] et décrite au chapitre précédent, basée sur une méthode d'interpolation polynomiale de Lagrange et reposant sur un schéma explicite saute-mouton du second ordre. L'évolution temporelle de l'erreur L^2 correspondante à chacun des cas considérés est représentée sur la figure 2.35. Comme pour toute méthode explicite, il s'agit d'un schéma numérique dont la stabilité est contrainte par une condition CFL, qui multipliée par la taille caractéristique du plus plus petit élément du maillage va déterminer le pas de temps maximum autorisé pour la simulation. Comme nous l'avons vu précédemment (voir le

tableau 2.4.2), le nombre CFL tend à diminuer lorsque le degré d'interpolation augmente, ce qui peut engendrer de fortes pertes de performances lors de l'étape d'enrichissement de l'ordre d'approximation sur des maillages donnés. Cette considération se fait particulièrement ressentir par exemple sur une grille 11×11 où l'on voit sur le tableau 2.6.20 que le passage de l'ordre 3 à l'ordre 4 entraine deux fois plus d'itérations et que le passage de l'ordre 4 à l'ordre 5 entraine presque cinq fois plus d'itérations. Ainsi, combiné au coût suplémentaire lié à des matrices élémentaires plus grandes, cet enrichissement de l'ordre d'approximation pour un maillage donnée provoque des dégradations de performances conséquentes (facteurs $\times 3.5$ entre les ordre 3 et 4 et $\times 7.7$ entre les ordre 4 et 5).

Les méthodes explicites souffrent donc de l'impact négatif qu'ont les maillages fortement raffinés sur les performances. Cela est également très perceptible sur le tableau 2.6.20, où l'on constate que pour un ordre d'approximation donné compris entre 1 et 4 et pour un maillage contenant au moins 200 éléments ($N = 11$), le fait de raffiner le maillage en quadruplant le nombre d'éléments entraîne un ralentissement considérable de la méthode par un facteur avoisinant $\times 8$ pour un gain en précision d'un facteur au plus $\times 4$. Pour les ordres d'approximation les plus élevés, sur des maillages contenant très peu d'éléments, l'amélioration de la précision se fait évidemment davantage ressentir, avec un facteur de $\times 9$ par exemple pour une méthode GDDT-$\mathbb{P}_6$ mais témoigne tout de même d'une dégradation du temps CPU. Il faut donc veiller à prendre en compte la montée en ordre de l'approximation de la solution avec précaution et à l'associer à des maillages raffinés en fonction, pour compenser les pénalités provoquées à la fois par des mailles trop petites ou par des ordres d'approximation très grands dotés d'une faible condition CFL.

Une fois ces considérations effectuées, l'effet bénéfique de l'enrichissement de l'ordre d'approximation sur des maillages de plus en plus grossiers est particulièrement visible. Nous pouvons observer par exemple, que pour des erreurs L^2 finales exactement égales et pour un même nombre d'itérations, le fait de passer d'une configuration associée à un maillage composé de 800 triangles ($N = 21$) et à une approximation polynomiale d'ordre 3, au profit d'un maillage 4 fois plus petit ($N = 11$) et d'une montée de l'approximation à l'ordre 4, permet d'accélérer la méthode d'un facteur $\times 2.4$. Il en est de même sur des maillages très grossiers avec des ordres d'interpolation élevés où le passage de l'ordre 5 sur un maillage constitué de 32 éléments ($N = 5$) à l'ordre supérieur sur un maillage là encore 4 fois plus petit ($N = 3$), composé de seulement 8 triangles, permet de diviser le temps CPU par un facteur $\times 2.4$, tout en donnant une erreur L^2 finale très semblable.

Pour la suite des résultats, nous allons donc nous concentrer sur les méthodes GDDT-$\mathbb{P}_5$ et GDDT-$\mathbb{P}_6$ combinées à ces maillages très grossiers car nous avons vu qu'elles garantissent, lorsque combinées à interpolation polynomiale de Lagrange, des résultats très précis en un minimum de temps. Les résultats de performance concernant les bases de fonctions de Bernstein, d'Ainsworth-Coyle, de Solin et de Sherwin-Karniadakis sont représentés dans les tableaux 2.6.21 à 2.6.24 et les courbes de l'évolution temporelle de l'erreur L^2 associées sont données sur les figures 2.36 à 2.39. Le constat le plus apparent est que la base de Lagrange et la base de Bernstein fournissent les meilleurs résultats et s'imposent alors comme étant les bases des méthodes les plus rapides. Toutefois, l'attention est portée ici sur les bases hiérarchiques et il apparaît très nettement avec une grosse marge de différence que la base de Sherwin-Karniadakis se révèle être la moins avantageuse. En revanche, les deux autres bases hiérachiques donnent des résultats très semblables et bien qu'un peu plus coûteuses que les bases de Lagrange ou de Bernstein, n'exploitent encore pas au niveau algorithmique tout leur potentiel, les rendant ainsi très prometteuses pour une future exploitation totale de leur caractère hiérarchique.

2.6.2 Courant source localisé dans l'espace libre

Le second problème test considéré est la propagation d'une onde provenant d'un courant source localisé de la forme :

$$J_z(x, y, t) = \delta(x - x_s, y - y_s) f(t), \tag{2.6.1}$$

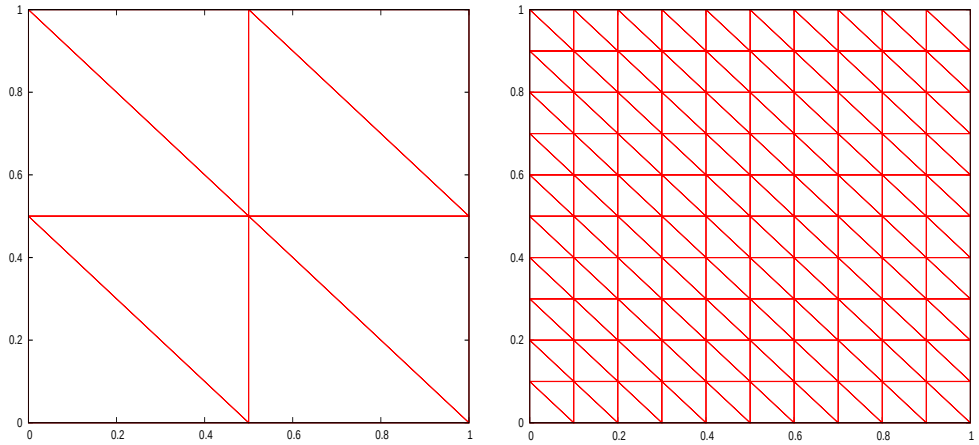


Figure 2.34 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : maillages triangulaires.

Interpolation	Maillage	# iter	Erreur L^2 finale	Temps CPU
$\mathbb{P}_1$	N=41	1698	8.07×10^{-3}	8.5 sec
-	N=81	3395	2.95×10^{-3}	68.2 sec
$\mathbb{P}_2$	N=21	1698	6.88×10^{-4}	3.4 sec
-	N=41	3395	1.69×10^{-4}	26.7 sec
$\mathbb{P}_3$	N=11	1415	9.09×10^{-4}	1.1 sec
-	N=21	2829	2.26×10^{-4}	9.1 sec
$\mathbb{P}_4$	N=11	2829	2.26×10^{-4}	3.8 sec
-	N=21	5657	5.61×10^{-5}	31.2 sec
$\mathbb{P}_5$	N=5	5657	5.30×10^{-5}	1.9 sec
-	N=11	14143	6.94×10^{-6}	29.3 sec
$\mathbb{P}_6$	N=3	5657	5.00×10^{-5}	0.8 sec
-	N=5	11314	5.55×10^{-6}	5.9 sec

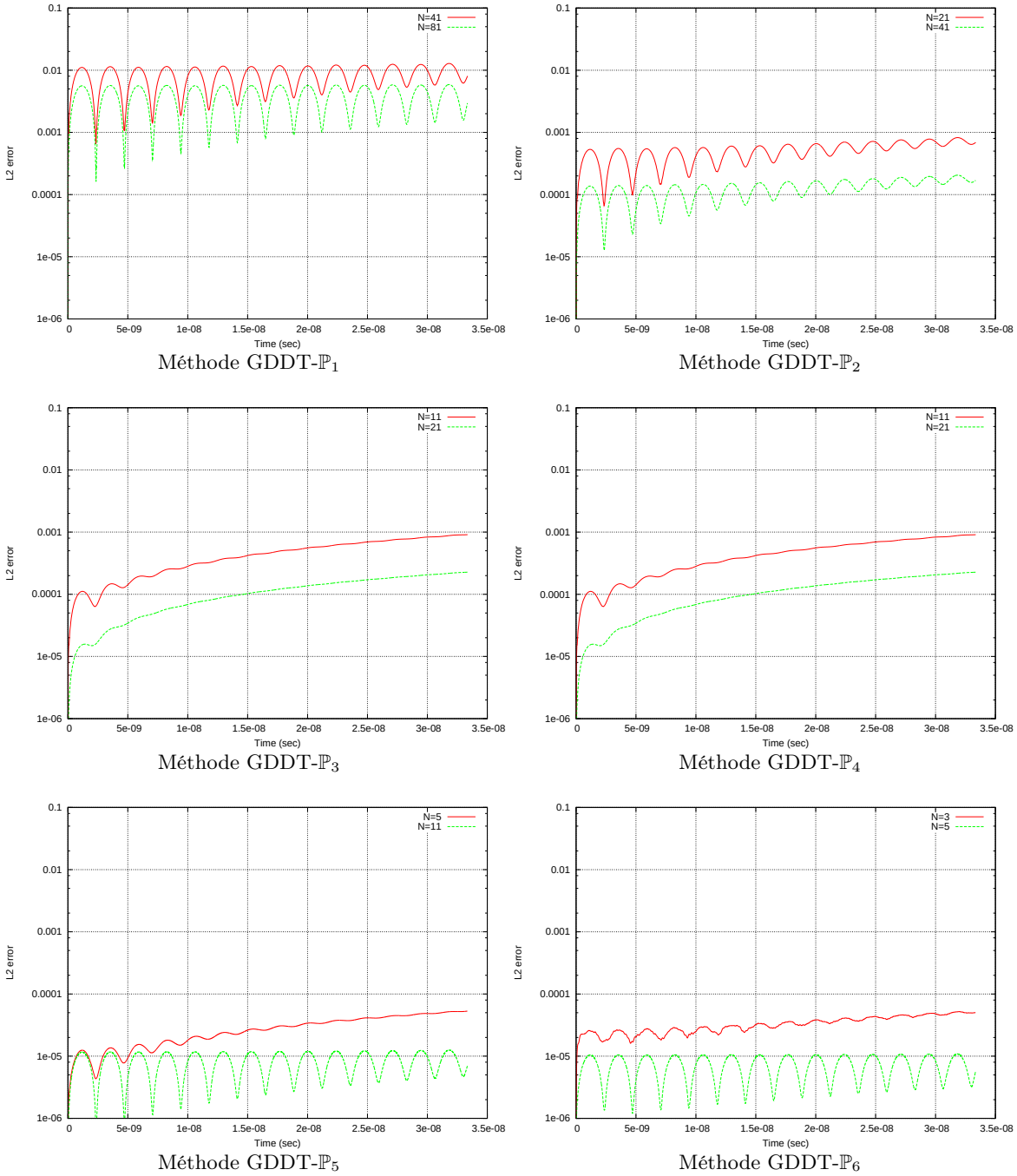
Table 2.6.20 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Lagrange. Résultats de performance (temps de simulation T=3.3×10^{-8} s).

Interpolation	Maillage	# iter	Erreur L^2 finale	Temps CPU
$\mathbb{P}_5$	N=5	5657	5.61×10^{-5}	1.9 sec
-	N=11	14143	8.17×10^{-6}	30.0 sec
$\mathbb{P}_6$	N=3	5657	5.69×10^{-5}	0.8 sec
-	N=5	11314	1.32×10^{-5}	5.9 sec

Table 2.6.21 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Bernstein. Résultats de performance (temps de simulation T=3.3×10^{-8} s).

Interpolation	Maillage	# iter	Erreur L^2 finale	Temps CPU
$\mathbb{P}_5$	N=5	5657	5.61×10^{-5}	2.7 sec
-	N=11	14143	8.17×10^{-6}	40.2 sec
$\mathbb{P}_6$	N=3	5657	5.69×10^{-5}	1.1 sec
-	N=5	11314	1.32×10^{-5}	8.3 sec

Table 2.6.22 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions d'Ainsworth-Coyle. Résultats de performance (temps de simulation T=3.3×10^{-8} s).



Méthode GDDT-$\mathbb{P}_1$ Méthode GDDT-$\mathbb{P}_2$

Méthode GDDT-$\mathbb{P}_3$ Méthode GDDT-$\mathbb{P}_4$

Méthode GDDT-$\mathbb{P}_5$ Méthode GDDT-$\mathbb{P}_6$

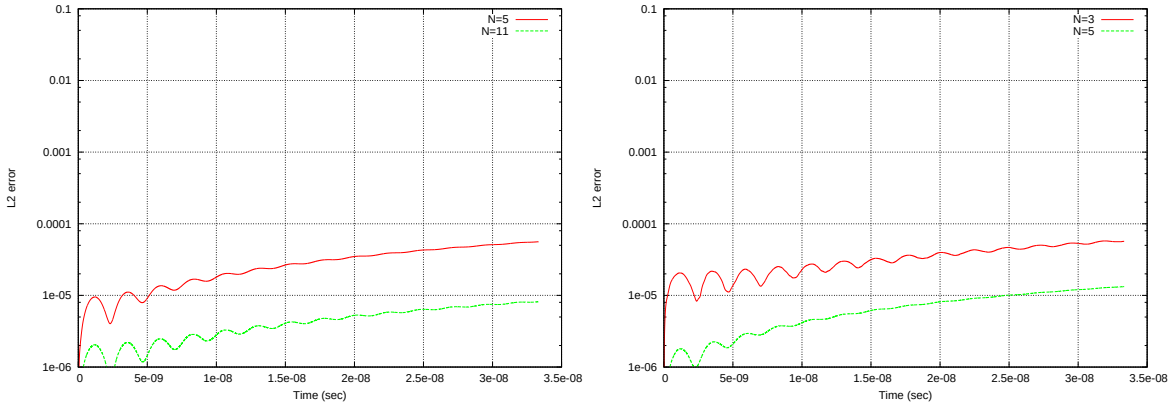
Figure 2.35 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Lagrange. Evolution temporelle de l'erreur L^2 (temps de simulation T=3.3×10^{-8} s).

Interpolation	Maillage	# iter	Erreur L^2 finale	Temps CPU
$\mathbb{P}_5$	N=5	5657	5.61×10^{-5}	2.6 sec
-	N=11	14143	8.17×10^{-6}	40.0 sec
$\mathbb{P}_6$	N=3	5657	5.69×10^{-5}	1.1 sec
-	N=5	11314	1.32×10^{-5}	8.4 sec

TABLE 2.6.23 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Solin. Résultats de performance (temps de simulation T=3.3×10^{-8} s).

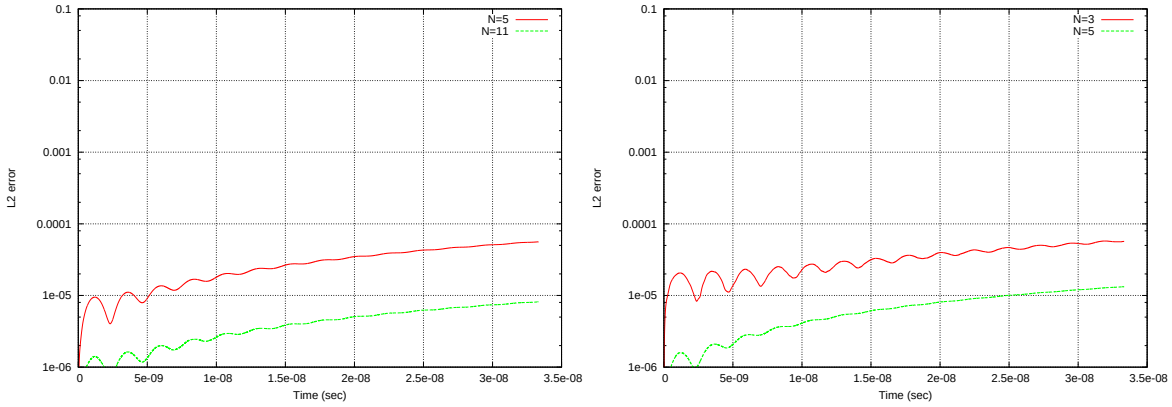
Interpolation	Maillage	# iter	Erreur L^2 finale	Temps CPU
$\mathbb{P}_5$	N=5	5657	5.61×10^{-5}	6.8 sec
-	N=11	14143	8.17×10^{-6}	113.5 sec
$\mathbb{P}_6$	N=3	5657	5.69×10^{-5}	2.4 sec
-	N=5	11314	1.32×10^{-5}	18.9 sec

TABLE 2.6.24 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Sherwin-Karniadakis. Résultats de performance (temps de simulation T=3.3×10^{-8} s).



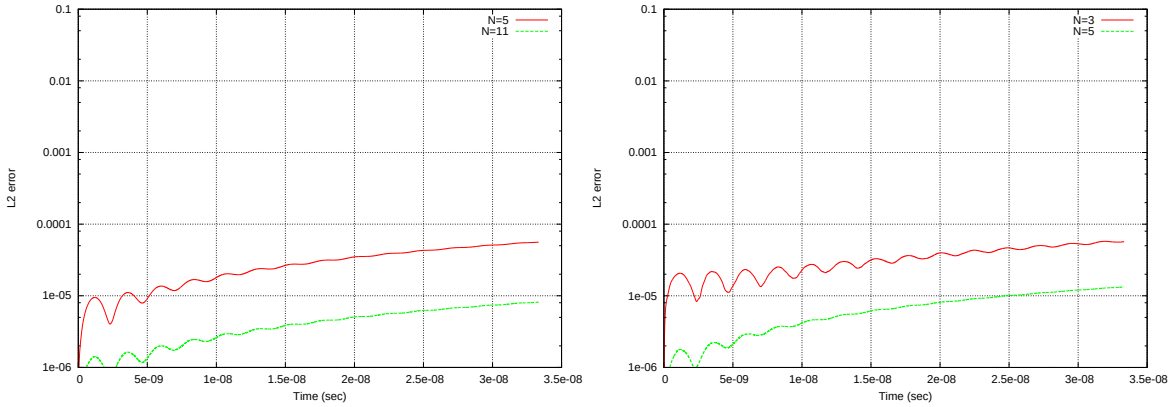
Méthode GDDT-$\mathbb{P}_5$ Méthode GDDT-$\mathbb{P}_6$

FIGURE 2.36 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Bernstein. Evolution temporelle de l'erreur L^2 (temps de simulation T=3.3×10^{-8} s).



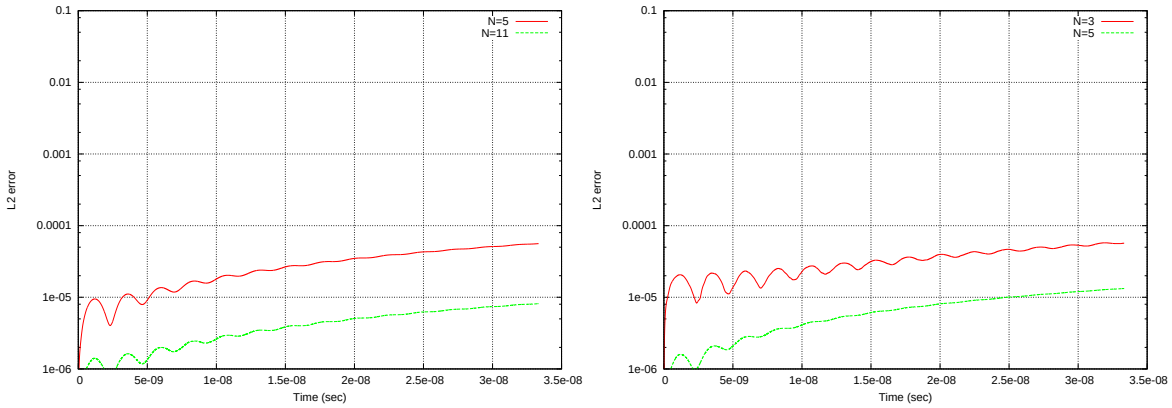
Méthode GDDT-$\mathbb{P}_5$ Méthode GDDT-$\mathbb{P}_6$

FIGURE 2.37 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions d'Ainsworth-Coyle. Evolution temporelle de l'erreur L^2 (temps de simulation T=3.3×10^{-8} s).



Méthode GDDT-$\mathbb{P}_5$ Méthode GDDT-$\mathbb{P}_6$

Figure 2.38 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Solin. Evolution temporelle de l'erreur L^2 (temps de simulation T=3.3×10^{-8} s).



Méthode GDDT-$\mathbb{P}_5$ Méthode GDDT-$\mathbb{P}_6$

Figure 2.39 – Mode propre dans une cavité carrée unitaire parfaitement conductrice : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Sherwin-Karniadakis. Evolution temporelle de l'erreur L^2 (temps de simulation T=3.3×10^{-8} s).

où $\delta(x-x_s, y-y_s)$ est la mesure de Dirac telle que $\delta(x-x_s, y-y_s) = 1$ si $x = x_s$ et $y = y_s$, 0 sinon. Pour le traitement numérique, une approche régularisée est adoptée qui consiste à remplacer la mesure de Dirac par une fonction Gaussienne. D'autre part, le signal temporel $f(t)$ est une fonction Gaussienne modulée par une fonction sinusoidale (voir la figure 2.40). La fréquence d'oscillation est fixée à 2 GHz. Le domaine de calcul est un carré unité et la condition absorbante de Silver-Müler est appliquée sur chaque côté du domaine. Comme dans le cas du problème test précédent, on utilise ici des maillages triangulaires uniformes obtenus par subdivision d'une grille cartésienne de N×N points. Le temps de simulation est fixé à $T = 5.10^{-8}$.

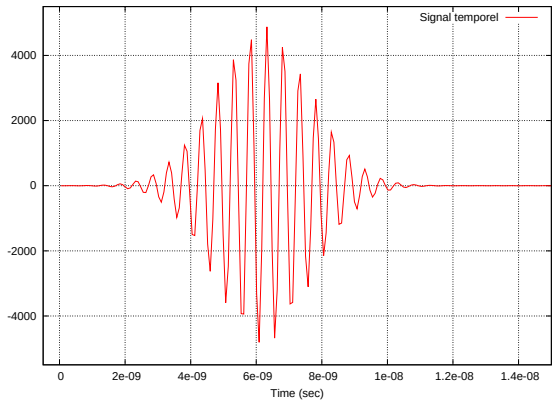


Figure 2.40 – Courant source localisé dans l'espace libre : signal temporel

2.6.2.1 Calculs en temps long

Avec cette première série de résultats effectués sur un maillage de $N = 11 \times 11$ points, on évalue l'impact du degré de l'approximation sur la convergence vers l'état stationnaire obtenu une fois que l'on a traversé la totalité du domaine de calcul. Le courant source est localisé au centre du domaine et on se place au point $[x = 0.6,\ y = 0.5]$ pour suivre l'évolution temporelle de l'énergie électromagnétique résiduelle discrète qui doit alors être nulle au terme de la simulation.

Les résultats concernant les méthodes GDDT-$\mathbb{P}_p$ d'ordre inférieur ou égal à 6, basées sur une base de fonctions de Lagrange sont présentés sur la figure 2.41 et renseignent de façon nette sur les ordres sujets aux meilleurs approximations. L'interpolation linéaire ($p = 1$) est de loin la plus mauvaise et ce constat se retrouve dans l'étude de l'évolution temporelle de la composante E_z du champ électromagnétique où grâce à la forme très oscillatoire du signal à propager, les figures 2.42 et 2.43 mettent en évidence des qualités très distinctes d'approximation de la solution numérique entre les méthodes GDDT d'ordre 1 et d'ordre 6. On remarque que l'amplitude du signal propagé est largement sous-estimée pour une interpolation linéaire comparativement à la méthode d'ordre 6 qui assure la disparition complète du courant au point d'observation au terme de la simulation alors que des oscillations d'amplitude importante persistent dans le cas de l'interpolation linéaire.

La figure 2.44 nous renseigne alors encore sur l'évolution temporelle de l'énergie électromagnétique discrète mais cette fois-ci pour notre échantillon de fonctions de bases, toutes directement représentées à l'ordre 6. Il ressort de cette analyse l'intérêt à utiliser pour ce problème test des bases modales ou hiérarchiques plutôt que la base nodale de Lagrange, qui semble plus sensible au faible raffinement du maillage. Cependant, la quasi correspondance des courbes relatives aux bases de Bernstein, Solin, Ainsworth-Coyle ou Sherkin-Karniadakis ne nous permet pas d'attester de la supériorité d'une base sur une autre en matière de précision des résultats. Dans ce qui suit, la base de Sherkin-Karniadakis n'est plus considérée car le cas test précédent du mode propre dans une cavité la place comme étant la moins rapide et, puisque les résultats obtenus montrent aussi des similarités de performance entre les deux autres bases hiérarchiques, nous portons notre attention sur la base de hiérarchique de Solin.

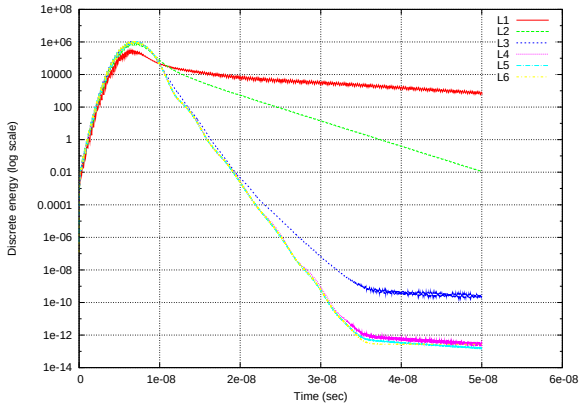
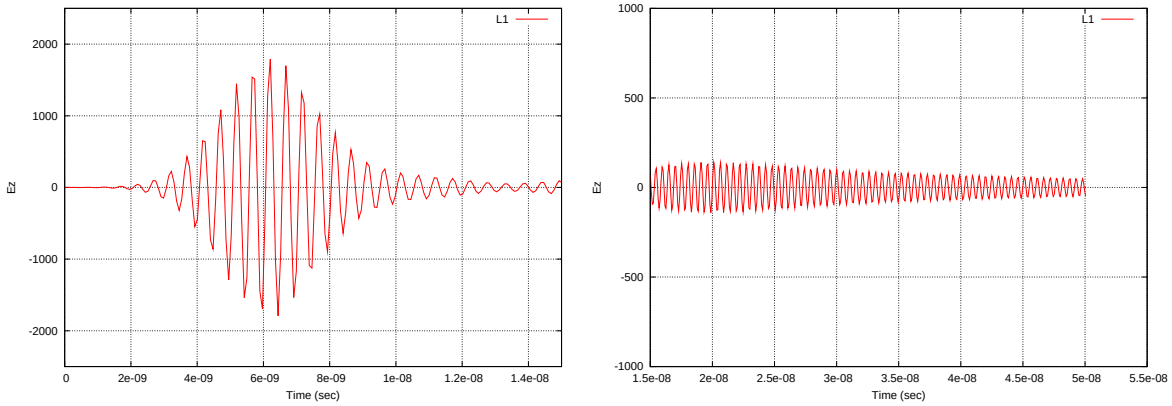
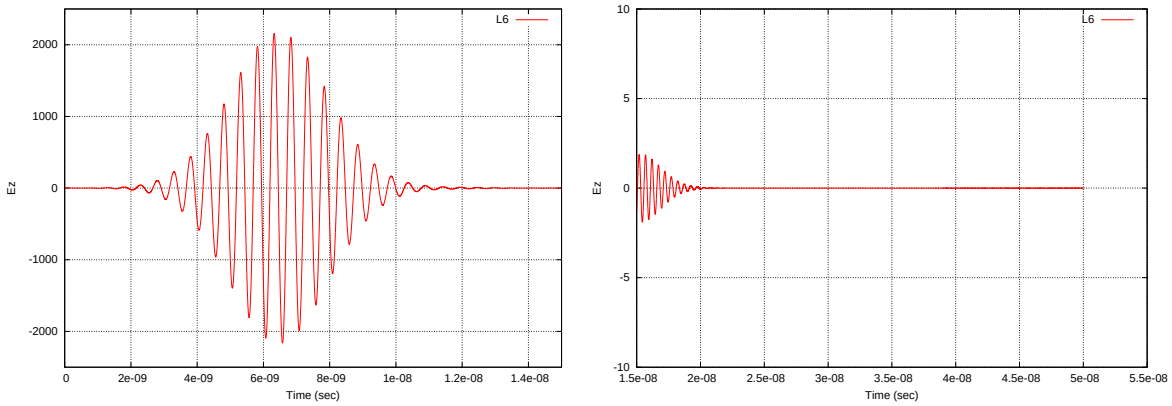


Figure 2.41 – Courant source localisé dans l'espace libre : méthode GDDT-$\mathbb{P}_p$ basée sur une base de fonctions de Lagrange. Evolution temporelle de l'énergie électromagnétique discrète pour $p = 1$ à 6. Maillage avec N=11×11 sommets.



Méthode GDDT-$\mathbb{P}_1$ Méthode GDDT-$\mathbb{P}_1$

Figure 2.42 – Courant source localisé dans l'espace libre : méthode GDDT-$\mathbb{P}_1$ basée sur une base de fonctions de Lagrange. Evolution temporelle de la composante E_z au point $[x = 0.6, y = 0.5]$. Maillage avec N=11 × 11 sommets.



Méthode GDDT-$\mathbb{P}_6$ Méthode GDDT-$\mathbb{P}_6$

Figure 2.43 – Courant source localisé dans l'espace libre : méthode GDDT-$\mathbb{P}_6$ basée sur une base de fonctions de Lagrange. Evolution temporelle de la composante E_z au point $[x = 0.6, y = 0.5]$. Maillage avec N=11 × 11 sommets.

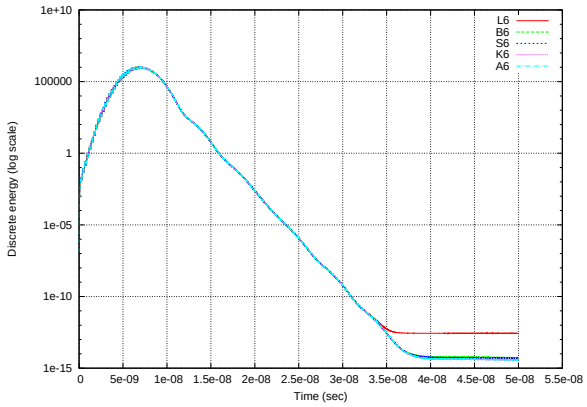
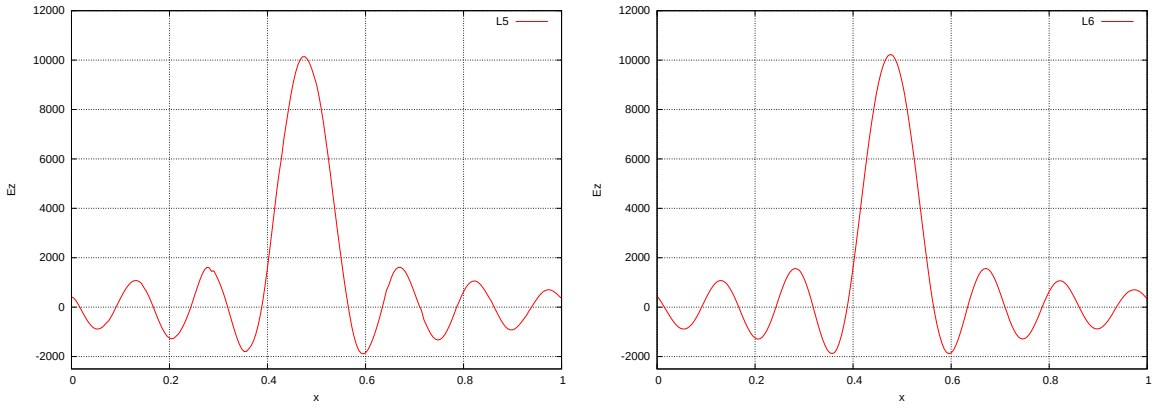


Figure 2.44 – Courant source localisé dans l'espace libre : méthode GDDT-$\mathbb{P}_6$. Evolution temporelle de l'énergie électromagnétique discrète pour différentes bases de fonctions. Maillage avec N=11 × 11 sommets.

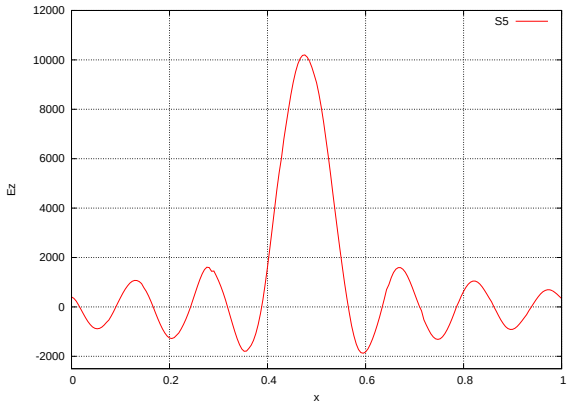
2.6.2.2 Influence combinée du pas de discrétisation et du degré d'interpolation

Le temps de simulation est maintenant fixé à T=6.6×10^{-9} s. On compare alors sur les figures 2.45 à 2.48 les distributions spatiales de la composante E_z suivant la droite $y = 0.5$ pour différentes valeurs du pas de discrétisation (i.e. du nombre de points N définissant la grille cartésienne) et du degré d'interpolation p pour des méthodes GDDT reposant sur des approximations polynomiales de Lagrange et de Solin. Pour un jeu de points de discrétisation du domaine composé de $N = 8 \times 8$ sommets, les figures 2.45 et 2.46 générées par un ordre d'approximation égal à 5 pour les deux bases présentent des défauts sur l'un des principaux sommets de la courbe qui disparaissent si l'on enrichit les méthodes d'un ordre supérieur où si l'on intensifie le raffinement du maillage, comme le montre la figure 2.47. Ce dernier phénomène se ressent particulièrement dans le cas d'approximations polynomiale de Lagrange d'ordres inférieurs, comme par exemple pour une méthode GDDT-$\mathbb{P}_4$ où l'on voit sur la figure 2.48 l'influence importante du pas de discrétisation sur la précision et le déphasage de la solution approchée.

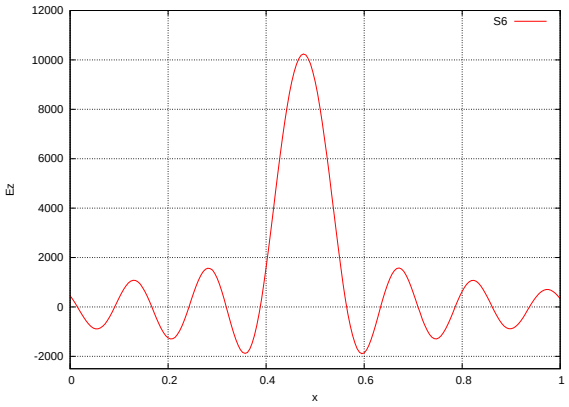


Méthode GDDT-$\mathbb{P}_5$ Méthode GDDT-$\mathbb{P}_6$

Figure 2.45 – Courant source localisé dans l'espace libre (F=2GHz) : méthodes GDDT-$\mathbb{P}_5$ et GDDT-$\mathbb{P}_6$ basées sur une base de fonctions de Lagrange. Distribution spatiale de la composante E_z suivant la droite $y = 0.5$. Maillage avec N=8 × 8 sommets.



Méthode GDDT-$\mathbb{P}_5$



Méthode GDDT-$\mathbb{P}_6$

Figure 2.46 – Courant source localisé dans l'espace libre (F=2GHz) : méthodes GDDT-$\mathbb{P}_5$ et GDDT-$\mathbb{P}_6$ basées sur une base de fonctions de Solin. Distribution spatiale de la composante E_z suivant la droite $y = 0.5$. Maillage avec N=8×8 sommets.

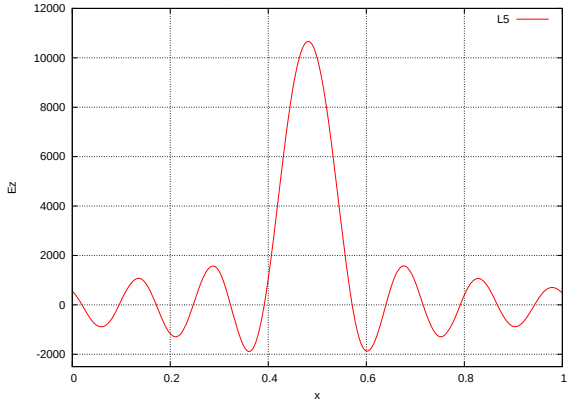


Figure 2.47 – Courant source localisé dans l'espace libre (F=2GHz) : méthode GDDT-$\mathbb{P}_5$ basée sur une base de fonctions de Lagrange. Distribution spatiale de la composante E_z suivant la droite $y = 0.5$. Maillage avec N=10×10 sommets.

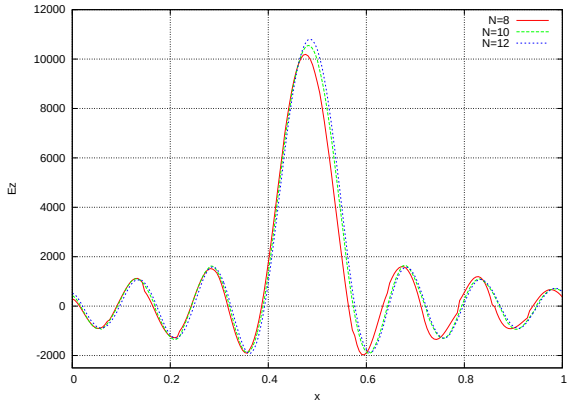


Figure 2.48 – Courant source localisé dans l'espace libre (F=2GHz) : méthode GDDT-$\mathbb{P}_4$ basée sur une base de fonctions de Lagrange. Distribution spatiale de la composante E_z suivant la droite $y = 0.5$.

2.7 Conclusion

A notre connaissance, aucune étude numérique comparative et détaillée de méthodes d'interpolation polynomiale n'avait été réalisée jusqu'ici dans le cadre de formulations Galerkin discontinues pour la résolution numérique des équations de Maxwell instationnaires sur des simplexes. Afin de tirer profit de la particularité de ces formulations à autoriser naturellement une non-conformité de l'approximation polynomiale, nous avons présenté dans ce chapitre une analyse exhaustive de différentes fonctions de bases locales polynomiales en dimensions 1 et 2, en association avec la méthode entièrement explicite GDDT-$\mathbb{P}_p$ décrite au premier chapitre et analysée dans [Fezoui 2005], comme une étude préalable à l'exploitation de la p-adaptivité pour la simulation numérique de problèmes de propagation tridimensionnels. La plupart des méthodes Galerkin discontinues présentes dans la littérature se basent sur une interpolation d'ordre élevé généralement construite à partir des polynômes de Lagrange. De par leur nature nodale, elles sont sujettes au phénomène de Runge et se révèlent peu efficaces et mal adaptées lorsque l'on souhaite faire varier la distribution des degrés de liberté sur les éléments du maillage. En revanche, les fonctions de base hiérarchiques orthogonales nécessitent moins de calculs pour le calcul des intégrales de volume et de surface, conduisent à des matrices locales mieux conditionnées et facilement déductibles les unes des autres lorsque l'ordre d'approximation est enrichi localement.

En dépit des différences notables entre les propriétés et les avantages propres à chaque type d'interpolation polynomiale (base nodale, modale ou hiérarchique), les problèmes instationnaires tests considérés en 1D et en 2D dans cette étude ne font ressortir aucune base en particulier concernant la précision des résultats, quelque soit le schéma d'intégration en temps utilisé. Il apparait donc que le rôle du conditionnement dans les méthodes utilisant un schéma explicite a très peu d'impact sur les résultats, même à l'ordre élevé, et reste faible comparativement aux méthodes implicites nécessitant la résolution d'un système linéaire à chaque pas de temps. Il serait néanmoins intéressant d'étendre l'étude de problèmes instationnaires à des ordres d'approximation supérieurs à 6 pour évaluer l'influence du conditionnement sur des matrices élémentaires de plus grande taille et éventuellement de distinguer plus nettement les bases entre elles selon leur précison.

L'étude numérique monodimensionnelle a fait ressortir l'intérêt à considérer des schémas d'intégration en temps d'ordre plus élevé que le schéma saute-mouton d'ordre 2. Les méthodes intégrées par des schémas en temps d'ordre 4 conduisent sans surprise à une meilleure précision numérique à condition toutefois d'utiliser le schéma approprié dans chacun des cas tests considérés pour lesquels la complexité du signal que l'on propage semble cruciale. Un avantage non négligeable des schémas en temps LF4 et RK4 est leur capacité à disposer de conditions CFL plus élevées que pour le schéma LF2, dictant ainsi un nombre réduit de pas de temps pour chacun des ordres d'approximation, indépendemment de la base polynomiale choisie. A priori, les bases se ne différencient pas les unes des autres sur le critère de stabilité mais une étude théorique de stabilité par une approche formelle est nécessaire afin d'écarter toute hypothèse pouvant être liée à la nature de la base d'interpolations polynomiale. En termes de performances, l'étude bidimensionnelle de problèmes instationnaires plus réalistes montre que les méthodes d'interpolation de Lagrange et de Bernstein sont les plus efficaces et que la base de Sherkin-Karniadakis est de loin la plus coûteuse. Les autres bases hiérarchiques, associées à des méthodes GDDT-$\mathbb{P}_5$ et GDDT-$\mathbb{P}_6$, se révèlent un peu plus lentes que les bases de Lagrange et de Bernstein pour les même ordres, toutefois cet écart de performance bien que non négligeable se fait moins ressentir pour des ordres d'approximation très élevés.

L'élaboration des méthodes GDDT-$\mathbb{P}_p$ reposant sur les fonctions de base hiérarchiques à partir de travaux existants n'a pas été aussi simple que prévu d'un point de vue algorithmique et a par conséquent modéré notre étude, qui demeure incomplète, mais qui offre toutefois un cadre préliminaire essentiel à la mise en œuvre de méthodes p-adaptives. D'autres contributions telles que l'étude de cas tests plus compliqués faisant par exemple intervenir des milieux hétérogènes, l'exploitation totale de la p-adaptivité d'un point de vue algorithmique ou encore l'étude de l'influence des bases sur les convergences numériques des méthodes GDDT-$\mathbb{P}_p$ associées peuvent également fournir de précieuses informations sur le choix d'un jeu optimal de fonctions de base et font partie des pistes restantes à explorer. L'objectif à plus long terme, en

continuité avec les travaux de cette thèse, est la construction et l'étude comparative d'estimateurs d'erreur a posteriori pour l'approximation des équations de Maxwell en domaine temporel en vue d'obtenir une méthodologie *hp*-adaptative pour la simulation numérique de problèmes de propagation tridimensionnels.

Chapitre 3

High order DGTD method with local time stepping

Sommaire

3.1 Introduction

The growing necessity to accurately and efficiently model wave dominated problems of increasing complexity is challenging current simulation tools. Large-scale applications related to acoustic, electromagnetic or elastic wave phenomena usually include the presence of complex geometry, realistic material properties and small geometrical features that require high-fidelity solution over locally refined meshes at a very high computational cost. Maxwell's equations can be discretized using several grid based methods. The first and probably most popular method, the Finite Difference Time Domain (FDTD) scheme [Yee 1966] is simple and efficient on structured grids, but its usefulness is somewhat limited on boundaries and interfaces, suffering from its inability to accurately model curved objects and small geometrical features. Indeed, the Cartesian FDTD grid leads to a staircase approximation of the geometry introducing an inaccurate representation of the solution [Cangellaris 1991] - [Taflove 2005]. Besides, FDTD methods are penalized by dispersive errors due to the centered nature of the approximations of the field spatial derivatives. In contrast with Yee's scheme, Finite Volume Time Domain (FVTD) methods [Bonnet 1997] - [Piperno 2002] - [Edelvik 2000] benefit from the strong additional advantage to easily handle unstructured grids and complex geometries using cells of any shape (tetrahedra, hexahedra, prisms, etc.) to allow spatial refinement for better performances both in computing time and accuracy. However, FVTD

methods often rely on upwind approximations and are thus too dissipative, which make them inappropriate candidates for long time wave computations where it is more efficient to increase the order of the polynomial approximation than refining the mesh (see [Hesthaven 2002] for details). A centered finite volume scheme with properties that are very similar to the ones of Yee's scheme has been proposed in [Piperno 2002]. This scheme can handle unstructured tetrahedral meshes however the inherent dispersion of the scheme avoid accurate simulations on highly non-uniform discretizations. As an alternative, discontinuous Galerkin (DG) methods, introduced by Reed and Hill [Reed 1973] for the scalar neutron transport equation, naturally allow high order spatial approximation of the field in each cell and offer even greater flexibility for local mesh refinement by accommodating non-conforming grids and hanging nodes [Hesthaven 2002] - [Piperno 2003] - [Cohen 2006]- [Fezoui 2005] - [Fahs 2009].

Most explicit time stepping methods are conditionally stable since stability is ensured by complying with the Courant-Friedrichs-Lewy (CFL) criterion, defined with respect to the time integration scheme in use. In a global scheme, the finest element in a non-uniform mesh dictates the maximum time step allowed to all the other elements of the computational domain. One idea developed in [Piperno 2006b] - [Dolean 2010] consists in using an implicit time integration scheme on a subset of the mesh elements formed with the smallest elements in order to exploit their inherent property of unconditional stability, and allow a larger time step be preserved on the whole domain. Although it shows interesting mathematical features, a method combining an implicit scheme with a general explicit method can be costly from the numerical point of view, depending on the size of the matrix to be inverted at each time step. Thus, such a locally implicit DGTD method is particularly well suited to situations for which the refined part of the underlying mesh is of small size as demonstrated in [Dolean 2010].

When mesh refinement is restricted to a small region, the use of a small time step in the entire computational domain severely increases time consumption and the ratio $c\,\Delta t/h$, where c denotes the propagation velocity, gets much smaller than its optimal value on the coarse grid, which generates dispersion errors. To overcome this bottleneck and reduce the cost of the simulation especially when large ratios of mesh size are of concern, various local explicit time stepping strategies have been proposed, presenting the advantage to use non-uniform time step sizes on non-uniform meshes to further improve the efficiency of the numerical scheme by setting time steps accordingly to their corresponding element size. In this case, one can have a non-conformity between the time grids and so the coupling between the sub-grids is more complicated. The first attempts in the electromagnetic literature were based on interpolation techniques e.g. [Chevalier 1997][Kunz 1981]. The basic idea is to use the interior scheme over all the computational domain obtaining some values of the solution at the interface between sub-grids with adequate interpolation formulas (in space or time) to render the scheme consistent. Unfortunately, the resulting schemes appear to be difficult to analyse and may be unstable under the usual CFL condition. In order to overcome these difficulties, Collino *et al.* have proposed a second order local time stepping method for the first-order wave equation [Collino 2003a]. This scheme has been analysed in [Collino 2003b]-[Joly 2005] (see also [Garcia 2004] for the encompassing analysis by Fourier like techniques) and extended to elastodynamics [Bécache 2005] and Maxwell's equations for a non-conducting media [Collino 2006]. Their approach, which is based on the introduction of a Lagrange multiplier, conserves a discrete energy that guarantees the stability a the numerical scheme. However it requires at every time step the solution of a linear system on the interface between the coarse and the fine parts of the mesh. Thus, for 3D Maxwell equations, this method leads to a numerical scheme too expensive in terms of memory and computational time.

Since DG methods are inherently local, they are particularly well suited for the development of explicit local time stepping schemes. By combining the symplectic Strömer-Verlet method with a DG discretization of Maxwell's equations in first-order form, Piperno [Piperno 2006a]-[Piperno 2006b] has proposed an explicit second order local time stepping scheme, which also conserves a discrete energy. In addition, this scheme permits to treat the entire problem as a set of cell areas for which a local time step is given. Then the evolution of the fields in time can be easily done by a recursive process. However, the requirement of field values at time stations that are not computed implies the loss of the symplectic pro-

perty of the scheme. In [Montseny 2008], Montseny *et al.* combined a similar recursive integrator with a DG formulation on hexahedral elements. Although hexahedral elements are very efficient, they can produce spurious modes while automated grid generation only with such elements remains a non-trivial task. Starting from the standard leap-frog scheme, Diaz and Grote [Diaz 2009] devised an explicit energy conserving local time stepping scheme of arbitrarily high accuracy for the homogeneous wave equation. Their objective (energy conservation) is close to [Ezziani 2010] but the method consists more in working on the time discretization, typically in the fine grid. Deriving from [Diaz 2009], Baldassari considered in her PhD thesis [Baldassari 2009] an Interior Penalty Discontinuous Galerkin method (IPDG) to simulate the propagation of acoustic waves in heterogeneous 2D and 3D media and combined a p-adaptivity in space with the application of a local time stepping strategy which permits not only the use of different time steps, but also to adapt the order of the time discretization to the order of each cells.

Recently, Taube *et al.* [Taube 2009] proposed an explicit local time stepping method for Maxwell's equations by extending the so-called arbitrary high-order derivatives (ADER) DG approach [Dumbser 2007] to Maxwell's equations. In this DG formulation, the solution is expanded in Taylor series in time and then the Cauchy-Kovalevskaya (CK) procedure, which is sometimes called the Lax-Wendroff procedure [Qui 2005], is used to replace the time derivatives in this series by space derivatives. In [Schomann 2010] a local time stepping algorithm based on Taylor expansion combined to a high order DG method is presented. The same authors also proposed in [Gödel 2009] a local time stepping scheme based on an Adams-Bashforth (AB) multistep method but they highlighted the important drawback of losing the Single Instruction Multiple Data (SIMD) characteristics in view of the implementation of the resulting Maxwell solver a graphics card accelerator. Another different approach consists in considering explicit high-order Runge-Kutta schemes with non-uniform time step sizes which makes the time integration of semi-discrete systems on non-uniform grid more efficient. In order to ensure correct communication of solutions on the interfaces of grids with different time step sizes, the values at intermediate stages of the Runge-Kutta time integration on the elements neighboring such interfaces are coupled with minimal dissipation and dispersion errors [Liu 2010].

In this study, we propose fully explicit high order local time stepping strategies in the spirit of those recently proposed in [Diaz 2009][Grote 2009]. This is done here in the framework of the non-dissipative discontinuous Galerkin time domain (DGTD) method [Fezoui 2005] for the solution of the first order form of the system of Maxwell's equations. The outline of this chapter is as follows. In section 3.2, we introduce the discontinuous Galerkin spatial discretization for Maxwell's equations and recall the centered leap-frog second order explicit global time stepping. In section 3.3, we propose a second order local time stepping method for the first order Maxwell's equations. The algorithmic aspect of the three steps scheme is presented in full details and elementary stability properties are proved. Numerical experiments in one and two dimensional cases are conducted to validate the theory and illustrate the usefulness of the local time stepping scheme in terms of computational benefits. We show how a small overlap between the fine and the coarse regions achieves an optimal CFL condition. Section 3.6 concludes this study and suggests future works.

This chapter has been the subject of a study undertaken in collaboration with S. Descombes (University of Nice-Sophia Antipolis and INRIA Sophia Antipolis - Méditerranée, NACHOS project-team) and J. Diaz (INRIA Bordeaux - Sud-Ouest, MAGIQUE-3D project-team).

3.2 DGTD method on tetrahedral meshes

3.2.1 Continuous problem

We consider the Maxwell's equations in three space dimensions for heterogeneous linear isotropic media. The electric field $\vec{E}(\vec{x},t) = {}^t(E_x, E_y, E_z)$ and the magnetic field $\vec{H}(\vec{x},t) = {}^t(H_x, H_y, H_z)$ verify :

$$\left\{ \begin{array}{lcl} \epsilon \partial_t \vec{E} & - & \mathrm{curl}\vec{H} = -\vec{J}, \\ \mu \partial_t \vec{H} & + & \mathrm{curl}\vec{E} = 0, \end{array} \right. \tag{3.2.1}$$

where the symbol ∂_t denotes a time derivative and $\vec{J}(\vec{x},t)$ is a current source term. These equations are set on a bounded polyhedral domain Ω of $\mathbb{R}^3$. The electric permittivity $\epsilon(\vec{x})$ and the magnetic permeability coefficients $\mu(\vec{x})$ are varying in space, time-invariant and both positive functions. The current source term $\vec{J}$ is the sum of the conductive current $\vec{J_\sigma} = \sigma\vec{E}$ (where $\sigma(\vec{x})$ denotes the electric conductivity of the media) and of an applied current $\vec{J_s}$ associated to a localized source for the incident electromagnetic field. Our goal is to solve system (3.2.1) in a domain Ω with boundary $\partial\Omega = \Gamma_a \cup \Gamma_m$, where we impose the following boundary conditions :

$$\left\{ \begin{array}{lcl} \vec{n} \times \vec{E} & = & 0 \text{ on } \Gamma_m, \\ \mathcal{L}(\vec{E}, \vec{H}) & = & \mathcal{L}(\vec{E}_{\mathrm{inc}}, \vec{H}_{\mathrm{inc}}) \text{ on } \Gamma_a, \end{array} \right. \tag{3.2.2}$$

where $\mathcal{L}(\vec{E},\vec{H}) = \vec{n}\times\vec{E} - \sqrt{\frac{\mu}{\varepsilon}}\vec{n}\times(\vec{H}\times\vec{n})$. Here $\vec{n}$ denotes the unit outward normal to $\partial\Omega$ and $(\vec{E}_{\mathrm{inc}}, \vec{H}_{\mathrm{inc}})$ is a given incident field. The first boundary condition is called *metallic* (referring to a perfectly conducting surface) while the second condition is called *absorbing* and takes here the form of the Silver-Müller condition which is a first order approximation of the exact absorbing boundary condition. This absorbing condition is applied on Γ_a which represents an artificial truncation of the computational domain. Finally, system (3.2.1) is supplemented with initial conditions : $\vec{E}_0(\vec{x}) = \vec{E}(\vec{x},t)$ and $\vec{H}_0(\vec{x}) = \vec{H}(\vec{x},t)$.

3.2.2 Discretization in space

We consider a partition $\mathcal{T}_h$ of Ω into a set of tetrahedra τ_i of size h_i with boundary $\partial\tau_i$ such that $h = \max_{\tau_i\in\mathcal{T}_h} h_i$. For each τ_i, V_i denotes its volume, and ϵ_i and μ_i are respectively the local electric permittivity and magnetic permeability of the medium, which are assumed constant inside the element τ_i. For two distinct tetrahedra τ_i and τ_k in $\mathcal{T}_h$, the intersection $\tau_i \cap \tau_k$ is a triangle a_{ik} which we will call interface. For each internal interface a_{ik}, we denote by S_{ik} the measure of a_{ik} and by $\vec{n}_{ik}$ the unitary normal vector, oriented from τ_i to τ_k. For the boundary interfaces, the index k corresponds to a fictitious element outside the domain. We denote by $\mathcal{F}_h^I$ the union of all interior interfaces of $\mathcal{T}_h$, by $\mathcal{F}_h^B$ the union of all boundary interfaces of $\mathcal{T}_h$, and by $\mathcal{F}_h = \mathcal{F}_h^I \cap \mathcal{F}_h^B$. Furthermore, we identify $\mathcal{F}_h^B$ to $\partial\Omega$ since Ω is a polyhedron. Finally, we denote by $\mathcal{V}_i$ the set of indices of the elements which are neighbors of τ_i (having an interface in common). In the following, to simplify the presentation, we set $\vec{J} = 0$. For a given partition $\mathcal{T}_h$, we seek approximate solutions to (3.2.1) in the finite dimensional subspace :

$$V_{p_i}(\mathcal{T}_h) = \{\vec{v} \in L^2(\Omega)^3 : \vec{v}_{|\tau_i} \in (\mathbb{P}_{p_i}(\tau_i))^3 \quad \forall \tau_i \in \mathcal{T}_h\}, \tag{3.2.3}$$

where $\mathbb{P}_{p_i}(\tau_i)$ denotes the space of nodal polynomial functions of degree at most p_i inside the element τ_i. Following the discontinuous Galerkin approach, the electric and magnetic fields inside each finite element are seek for as linear combinations $(\vec{\mathbf{E}}_i, \vec{\mathbf{H}}_i)$ of linearly independent basis vector fields $\vec{\varphi}_{ij}$, $1 \le j \le d_i$, where d_i denotes the local number of degrees of freedom inside τ_i. Let $\mathcal{P}_i = \mathrm{Span}(\vec{\varphi}_{ij},\ 1 \le j \le d_i)$. The approximate fields $(\vec{\mathbf{E}}_h, \vec{\mathbf{H}}_h)$, defined by $(\forall i, \vec{\mathbf{E}}_{h|\tau_i} = \vec{\mathbf{E}}_i, \vec{\mathbf{H}}_{h|\tau_i} = \vec{\mathbf{H}}_i)$ are allowed to be completely discontinuous across element boundaries. For such a discontinuous field $\vec{\mathbf{U}}_h$, we define its average $\{\vec{\mathbf{U}}_h\}_{ik}$

through any internal interface a_{ik}, as $\{\vec{\mathbf{U}}_h\}_{ik} = (\vec{\mathbf{U}}_{i|a_{ik}} + \vec{\mathbf{U}}_{k|a_{ik}})/2$. Note that for any internal interface a_{ik}, $\{\vec{\mathbf{U}}_h\}_{ki} = \{\vec{\mathbf{U}}_h\}_{ik}$. Because of this discontinuity, a global variational formulation cannot be obtained. However, dot-multiplying (3.2.1) by any given vector function $\vec{\varphi} \in \mathcal{P}_i$, integrating over each single element τ_i and integrating by parts, yields :

$$\left\{ \begin{array}{rcl} \displaystyle\int_{\tau_i} \vec{\varphi} \cdot \epsilon_i \partial_t \vec{\mathbf{E}} & = & \displaystyle\int_{\tau_i} \mathrm{curl}\vec{\varphi} \cdot \vec{\mathbf{H}} - \int_{\partial\tau_i} \vec{\varphi} \cdot (\vec{\mathbf{H}} \times \vec{n}), \\ \displaystyle\int_{\tau_i} \vec{\varphi} \cdot \mu_i \partial_t \vec{\mathbf{H}} & = & \displaystyle -\int_{\tau_i} \mathrm{curl}\vec{\varphi} \cdot \vec{\mathbf{E}} + \int_{\partial\tau_i} \vec{\varphi} \cdot (\vec{\mathbf{E}} \times \vec{n}). \end{array} \right. \tag{3.2.4}$$

In Eq. (3.2.4), we now replace the exact fields $\vec{\mathbf{E}}$ and $\vec{\mathbf{H}}$ by the approximate fields $\vec{\mathbf{E}}_h$ and $\vec{\mathbf{H}}_h$ in order to evaluate volume integrals. For integrals over $\partial\tau_i$, a specific treatment must be introduced since the approximate fields are discontinuous through element faces, leading to the definition of a *numerical flux*. We choose to use a fully centered numerical flux, *i.e.* $\forall i, \forall k \in \mathcal{V}_i$, $\vec{\mathbf{E}}_{|a_{ik}} \simeq \{\vec{\mathbf{E}}_h\}_{ik}$, $\vec{\mathbf{H}}_{|a_{ik}} \simeq \{\vec{\mathbf{H}}_h\}_{ik}$. The metallic boundary condition (first relation of (3.2.2)) on a boundary interface $a_{ik} \in \Gamma_m$ (k in the element index of the fictitious neighboring element) is dealt with *weakly*, in the sense that traces of fictitious fields $\vec{\mathbf{E}}_k$ and $\vec{\mathbf{H}}_k$ are used for the computation of numerical fluxes for the boundary element τ_i. More precisely, we set $\vec{\mathbf{E}}_{k|a_{ik}} = -\vec{\mathbf{E}}_{i|a_{ik}}$ and $\vec{\mathbf{H}}_{k|a_{ik}} = \vec{\mathbf{H}}_{i|a_{ik}}$. Similarly, the absorbing boundary condition (second relation of (3.2.2)) is taken into account through the use of a fully upwind numerical flux for the evaluation of the corresponding boundary integral over $a_{ik} \in \Gamma_a$ (see [Catella 2010] for more details). Evaluating the surface integrals in (3.2.4) using the centered numerical flux, and re-integrating by parts yields :

$$\left\{ \begin{array}{rcl} \displaystyle\int_{\tau_i} \vec{\varphi} \cdot \epsilon_i \partial_t \vec{\mathbf{E}}_i & = & \displaystyle\frac{1}{2}\int_{\tau_i} (\mathrm{curl}\vec{\varphi} \cdot \vec{\mathbf{H}}_i + \mathrm{curl}\vec{\mathbf{H}}_i \cdot \vec{\varphi}) - \frac{1}{2}\sum_{k\in\mathcal{V}_i} \int_{a_{ik}} \vec{\varphi} \cdot (\vec{\mathbf{H}}_k \times \vec{n}_{ik}), \\ \displaystyle\int_{\tau_i} \vec{\varphi} \cdot \mu_i \partial_t \vec{\mathbf{H}}_i & = & \displaystyle -\frac{1}{2}\int_{\tau_i} (\mathrm{curl}\vec{\varphi} \cdot \vec{\mathbf{E}}_i + \mathrm{curl}\vec{\mathbf{E}}_i \cdot \vec{\varphi}) + \frac{1}{2}\sum_{k\in\mathcal{V}_i} \int_{a_{ik}} \vec{\varphi} \cdot (\vec{\mathbf{E}}_k \times \vec{n}_{ik}). \end{array} \right. \tag{3.2.5}$$

Eq. (3.2.5) can be rewritten in terms of scalar unknowns. Inside each element, the fields are re-composed according to $\vec{\mathbf{E}}_i = \sum_{1\le j\le d} E_{ij}\vec{\varphi}_{ij}$ and $\vec{\mathbf{H}}_i = \sum_{1\le j\le d} H_{ij}\vec{\varphi}_{ij}$ and let us now denote by $\mathbf{E}_i$ and $\mathbf{H}_i$ respectively the column vectors $(E_{il})_{1\le l\le d_i}$ and $(H_{il})_{1\le l\le d_i}$. Then, (3.2.5) is equivalent to :

$$\left\{ \begin{array}{rcl} \displaystyle M_i^{\epsilon}\frac{d\mathbf{E}_i}{dt} & = & \displaystyle K_i\mathbf{H}_i - \sum_{k\in\mathcal{V}_i} S_{ik}\mathbf{H}_k, \\ \displaystyle M_i^{\mu}\frac{d\mathbf{H}_i}{dt} & = & \displaystyle -K_i\mathbf{E}_i + \sum_{k\in\mathcal{V}_i} S_{ik}\mathbf{E}_k, \end{array} \right. \tag{3.2.6}$$

where the symmetric positive definite mass matrices M_i^{η} (η stands for ϵ or μ), the symmetric stiffness matrix K_i (both of size $d_i \times d_i$) and the symmetric interface matrix S_{ik} (of size $d_i \times d_j$ in the general case) are given by :

$$\begin{array}{rcl} (M_i^{\eta})_{jl} & = & \eta_i \displaystyle\int_{\tau_i} {}^t\vec{\varphi}_{ij} \cdot \vec{\varphi}_{il}, \\ (K_i)_{jl} & = & \displaystyle\frac{1}{2}\int_{\tau_i} {}^t\vec{\varphi}_{ij} \cdot \mathrm{curl}\vec{\varphi}_{il} + {}^t\vec{\varphi}_{il} \cdot \mathrm{curl}\vec{\varphi}_{ij}, \\ (S_{ik})_{jl} & = & \displaystyle\frac{1}{2}\int_{a_{ik}} {}^t\vec{\varphi}_{ij} \cdot (\vec{\varphi}_{kl} \times \vec{n}_{ik}). \end{array}$$

The set of local system of ordinary differential equations for each τ_i (3.2.6) can be formally transformed in a global system. To this end, we suppose that all electric (resp. magnetic) unknowns are gathered in

a column vector $\mathbb{E}$ (resp. $\mathbb{H}$) of size $d_g = \sum_{i=1}^{N_t} d_i$ where N_t stands for the number of elements in $\mathcal{T}_h$. Then system (3.2.6) can be rewritten as :

$$\left\{ \begin{array}{rcl} \mathbb{M}^\epsilon \dfrac{d\mathbb{E}}{dt} & = & \mathbb{K}\mathbb{H} - \mathbb{A}\mathbb{H} - \mathbb{B}\mathbb{H} + \mathbb{C}_E\mathbb{E}, \\ \mathbb{M}^\mu \dfrac{d\mathbb{H}}{dt} & = & -\mathbb{K}\mathbb{E} + \mathbb{A}\mathbb{E} - \mathbb{B}\mathbb{E} + \mathbb{C}_H\mathbb{H}, \end{array} \right. \tag{3.2.7}$$

where we have the following definitions and properties :

- $\mathbb{M}^\epsilon, \mathbb{M}^\mu$ and $\mathbb{K}$ are $d_g \times d_g$ block diagonal matrices with diagonal blocks equal to M_i^ϵ, M_i^μ and K_i respectively. $\mathbb{M}^\epsilon$ and $\mathbb{M}^\mu$ are symmetric positive definite matrices, and $\mathbb{K}$ is a symmetric matrix.
- $\mathbb{A}$ is also a $d_g \times d_g$ block sparse matrix, whose non-zero blocks are equal to S_{ik} when $a_{ik} \in \mathcal{F}_h^I$. Since $\vec{n}_{ki} = -\vec{n}_{ik}$, it can be checked that $(S_{ik})_{jl} = (S_{ki})_{lj}$ and then $S_{ki} = {}^tS_{ik}$; thus $\mathbb{A}$ is a symmetric matrix.
- $\mathbb{B}$ is a $d_g \times d_g$ block diagonal matrix, whose non-zero blocks are equal to S_{ik} when $a_{ik} \in \mathcal{F}_h^B \cap \Gamma_m$. In that case, $(S_{ik})_{jl} = -(S_{ik})_{lj}$; thus $\mathbb{B}$ is a skew-symmetric matrix.
- $\mathbb{C}_E$ and $\mathbb{C}_H$ are $d_g \times d_g$ block diagonal matrices associated to boundary integral terms for $a_{ik} \in \mathcal{F}_h^B \cap \Gamma_a$.

Consequently, if we set $\mathbb{S} = \mathbb{K} - \mathbb{A} - \mathbb{B}$, the system (3.2.7) rewrites as :

$$\left\{ \begin{array}{rcl} \mathbb{M}^\epsilon \dfrac{d\mathbb{E}}{dt} & = & \mathbb{S}\mathbb{H} + \mathbb{C}_E\mathbb{E}, \\ \mathbb{M}^\mu \dfrac{d\mathbb{H}}{dt} & = & -{}^t\mathbb{S}\mathbb{E} + \mathbb{C}_H\mathbb{H}. \end{array} \right. \tag{3.2.8}$$

3.3 Second order explicit local time stepping strategy

3.3.1 Formulation

We start from system (3.2.8) and assume for the moment that $\Gamma_a = \emptyset$:

$$\left\{ \begin{array}{rcl} \mathbb{M}^\epsilon \dfrac{d\mathbb{E}}{dt} & = & \mathbb{S}\mathbb{H}, \\ \mathbb{M}^\mu \dfrac{d\mathbb{H}}{dt} & = & -{}^t\mathbb{S}\mathbb{E}. \end{array} \right. \tag{3.3.1}$$

Since $\mathbb{M}^\epsilon$ and $\mathbb{M}^\mu$ are block diagonal matrices we can easily invert these matrices and rewrite system (3.3.1) as :

$$\left\{ \begin{array}{rcl} \dfrac{d\mathbb{E}}{dt} & = & \mathbb{S}^\epsilon\mathbb{H}, \\ \dfrac{d\mathbb{H}}{dt} & = & \mathbb{S}^\mu\mathbb{E}, \end{array} \right. \tag{3.3.2}$$

with $\mathbb{S}^\epsilon = (\mathbb{M}^\epsilon)^{-1}\mathbb{S}$ and $\mathbb{S}^\mu = -(\mathbb{M}^\mu)^{-1}\,{}^t\mathbb{S}$. We now assume a splitting of $\mathbb{E}$ and $\mathbb{H}$ as :

$$\left\{ \begin{array}{rcccl} \mathbb{E}(t) & = & (\mathbb{I}-\mathbb{P})\mathbb{E}(t) + \mathbb{P}\mathbb{E}(t) & = & \mathbb{E}^{[\text{coarse}]}(t) + \mathbb{E}^{[\text{fine}]}(t), \\ \mathbb{H}(t) & = & (\mathbb{I}-\mathbb{P})\mathbb{H}(t) + \mathbb{P}\mathbb{H}(t) & = & \mathbb{H}^{[\text{coarse}]}(t) + \mathbb{H}^{[\text{fine}]}(t), \end{array} \right. \tag{3.3.3}$$

where $\mathbb{I}$ is the identity matrix and $\mathbb{P}$ a projection matrix which is diagonal and with diagonal entries equal to zero or one, identifying the unknowns associated with the locally refined region of the mesh, where small time steps are needed. Following the approach adopted in [Diaz 2009][Grote 2009], we treat

${}^t(\mathbb{E}^{[\text{fine}]}(t), \mathbb{H}^{[\text{fine}]}(t))$ differently from ${}^t(\mathbb{E}^{[\text{coarse}]}(t), \mathbb{H}^{[\text{coarse}]}(t))$. For that purpose, introducing the auxiliary variable $\tau \in [0\ ,\ \Delta t]$, we consider system (3.3.2) from t^n to $t^n + \Delta t$:

$$\left\{ \begin{array}{rcl} \dfrac{d\mathbb{E}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\epsilon \left[(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n+\tau) + \mathbb{P}\mathbb{H}(t^n+\tau)\right], \quad \tau \in [0\ ,\ \Delta t], \\ \dfrac{d\mathbb{H}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\mu \left[(\mathbb{I}-\mathbb{P})\mathbb{E}(t^n+\tau) + \mathbb{P}\mathbb{E}(t^n+\tau)\right], \quad \tau \in [0\ ,\ \Delta t], \end{array} \right. \tag{3.3.4}$$

where Δt is a reference time step that will be specified later. This system can be rewritten as :

$$\left\{ \begin{array}{rcl} \dfrac{d\mathbb{E}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\epsilon \left[\mathbb{H}^{[\text{coarse}]}(t^n+\tau) + \mathbb{P}\mathbb{H}(t^n+\tau)\right], \quad \tau \in [0\ ,\ \Delta t], \\ \dfrac{d\mathbb{H}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\mu \left[\mathbb{E}^{[\text{coarse}]}(t^n+\tau) + \mathbb{P}\mathbb{E}(t^n+\tau)\right], \quad \tau \in [0\ ,\ \Delta t], \end{array} \right. \tag{3.3.5}$$

Next, we suppose that the variations of $\mathbb{H}^{[\text{coarse}]}(t+\tau)$ and $\mathbb{E}^{[\text{coarse}]}(t+\tau)$ are *small* enough (they are sometimes referred as the *slow components*) to consider the following approximations :

$$\left\{ \begin{array}{lll} \mathbb{H}^{[\text{coarse}]}(t^n+\tau) & \text{is approximated by} & \left| \begin{array}{ll} \mathbb{H}^{[\text{coarse}]}(t^n) & \text{for } \tau \in [0\ ,\ \dfrac{\Delta t}{2}], \\ \mathbb{H}^{[\text{coarse}]}(t^n+\Delta t) & \text{for } \tau \in [\dfrac{\Delta t}{2}\ ,\ \Delta t]. \end{array} \right. \\ \mathbb{E}^{[\text{coarse}]}(t^n+\tau) & \text{is approximated by} & \mathbb{E}^{[\text{coarse}]}(t^n+\dfrac{\Delta t}{2}) \quad \text{for } \tau \in [0\ ,\ \Delta t] \end{array} \right. \tag{3.3.6}$$

Now, for time advancing the field components $\mathbb{H}^{[\text{fine}]}(t)$ and $\mathbb{E}^{[\text{fine}]}(t)$ (that is, the *fast components*), we need now to introduce appropriate approximations for $\mathbb{E}^{[\text{coarse}]}(t^n + \dfrac{\Delta t}{2})$ and $\mathbb{H}^{[\text{coarse}]}(t^n + \Delta t)$. This can be done in several ways and we can for instance consider the following approximations :

$$\left\{ \begin{array}{rcl} \mathbb{E}^{[\text{coarse}]}(t^n+\dfrac{\Delta t}{2}) & = & (\mathbb{I}-\mathbb{P})\mathbb{E}(t^n+\dfrac{\Delta t}{2}) \\ & \approx & (\mathbb{I}-\mathbb{P})\left[\mathbb{E}(t^n) + \dfrac{\Delta t}{2}\dfrac{d\mathbb{E}}{d\tau}(t^n)\right] \\ & = & (\mathbb{I}-\mathbb{P})\left[\mathbb{E}(t^n) + \dfrac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}(t^n)\right], \\ \mathbb{H}^{[\text{coarse}]}(t^n+\Delta t) & = & (\mathbb{I}-\mathbb{P})\mathbb{H}(t^n+\Delta t) \\ & \approx & (\mathbb{I}-\mathbb{P})\left[\mathbb{H}(t^n) + \Delta t\dfrac{d\mathbb{H}}{d\tau}(t^n+\dfrac{\Delta t}{2})\right] \\ & = & (\mathbb{I}-\mathbb{P})\left[\mathbb{H}(t^n) + \Delta t\mathbb{S}^\mu\mathbb{E}(t^n+\dfrac{\Delta t}{2})\right] \\ & \approx & (\mathbb{I}-\mathbb{P})\left[\mathbb{H}(t^n) + \Delta t\left(\mathbb{S}^\mu\mathbb{E}^{[\text{coarse}]}(t^n+\dfrac{\Delta t}{2}) + \right.\right. \\ & & \left.\left. \mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(t^n+\dfrac{\Delta t}{2})\right)\right] \\ & \approx & (\mathbb{I}-\mathbb{P})\left[\mathbb{H}(t^n) + \Delta t\left(\mathbb{S}^\mu\left((\mathbb{I}-\mathbb{P})\left[\mathbb{E}(t^n) + \dfrac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}(t^n)\right]\right) + \right.\right. \\ & & \left.\left. \mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(t^n+\dfrac{\Delta t}{2})\right)\right]. \end{array} \right. \tag{3.3.7}$$

Here, the approximation of $\mathbb{H}(t^n + \Delta t)$ is computed from :

$$\frac{d\mathbb{H}}{d\tau}(t^n+\frac{\Delta t}{2}) \approx \frac{\mathbb{H}(t^n+\Delta t) - \mathbb{H}(t^n)}{\Delta t}.$$

In addition, the approximation of $\mathbb{E}^{[\text{coarse}]}(t^n + \frac{\Delta t}{2})$ given by the first relation of (3.3.7) has been exploited. Alternatively, we can use :

$$\left\{\begin{array}{rcl@{\qquad}rcl} \mathbb{E}^{[\text{coarse}]}(t^n + \frac{\Delta t}{2}) & = & (\mathbb{I}-\mathbb{P})\mathbb{E}(t^n + \frac{\Delta t}{2}) & \approx & (\mathbb{I}-\mathbb{P})\left[\mathbb{E}(t^n) + \frac{\Delta t}{2}\frac{d\mathbb{E}}{d\tau}(t^n)\right] \\ & & & = & (\mathbb{I}-\mathbb{P})\left[\mathbb{E}(t^n) + \frac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}(t^n)\right], \\ \mathbb{H}^{[\text{coarse}]}(t^n + \Delta t) & = & (\mathbb{I}-\mathbb{P})\mathbb{H}(t^n + \Delta t) & \approx & (\mathbb{I}-\mathbb{P})\left[\mathbb{H}(t^n) + \Delta t\frac{d\mathbb{H}}{d\tau}(t^n + \frac{\Delta t}{2})\right] \\ & & & = & (\mathbb{I}-\mathbb{P})\left[\mathbb{H}(t^n) + \Delta t\mathbb{S}^\mu\mathbb{E}(t^n + \frac{\Delta t}{2})\right], \end{array}\right. \tag{3.3.8}$$

which makes use of the vector $\mathbb{E}(t^n + \frac{\Delta t}{2})$ directly instead of its decomposition in coarse and fine parts. Using the approximations (3.3.7), we finally obtain the three steps scheme :

$$\left\{\begin{array}{rcl@{\qquad}l} \frac{d\mathbb{E}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\epsilon\left[(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n) + \mathbb{P}\mathbb{H}(t^n+\tau)\right], & \tau \in [0\ ,\ \frac{\Delta t}{2}], \\ \frac{d\mathbb{H}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\mu\left[(\mathbb{I}-\mathbb{P})\left(\mathbb{E}(t^n) + \frac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}(t^n)\right) + \mathbb{P}\mathbb{E}(t^n+\tau)\right], & \tau \in [0\ ,\ \Delta t], \\ \frac{d\mathbb{E}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\epsilon\left[(\mathbb{I}-\mathbb{P})\left(\mathbb{H}(t^n) + \Delta t\left(\mathbb{S}^\mu\left((\mathbb{I}-\mathbb{P})\left[\mathbb{E}(t^n) + \frac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}(t^n)\right]\right) + \right.\right.\right. & \\ & & \left.\left.\left.\mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(t^n + \frac{\Delta t}{2})\right)\right) + & \\ & & \left.\mathbb{P}\mathbb{H}(t^n+\tau)\right], \quad \tau \in [\frac{\Delta t}{2}\ ,\ \Delta t], & \end{array}\right. \tag{3.3.9}$$

or, for the alternative approximations (3.3.8) :

$$\left\{\begin{array}{rcll} \frac{d\mathbb{E}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\epsilon\left[(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n) + \mathbb{P}\mathbb{H}(t^n+\tau)\right], & \tau \in [0\ ,\ \frac{\Delta t}{2}], \\ \frac{d\mathbb{H}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\mu\left[(\mathbb{I}-\mathbb{P})\left(\mathbb{E}(t^n) + \frac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}(t^n)\right) + \mathbb{P}\mathbb{E}(t^n+\tau)\right], & \tau \in [0\ ,\ \Delta t], \\ \frac{d\mathbb{E}}{d\tau}(t^n+\tau) & = & \mathbb{S}^\epsilon\left[(\mathbb{I}-\mathbb{P})\left(\mathbb{H}(t^n) + \Delta t\mathbb{S}^\mu\mathbb{E}(t^n + \frac{\Delta t}{2})\right) + \right. & \\ & & \left.\mathbb{P}\mathbb{H}(t^n+\tau)\right], & \tau \in [\frac{\Delta t}{2}\ ,\ \Delta t]. \end{array}\right. \tag{3.3.10}$$

Then, the electromagnetic components are time advanced by integrating (3.3.9) or (3.3.10) for $\tau \in [0\ ,\ \Delta t]$ using the classical leap-frog scheme.

Remarks.

- An alternative strategy that could be considered is to first time advanced simultaneously the first and the second relations of (3.3.4) between 0 and $\frac{\Delta t}{2}$, and then time advanced simultaneously the first and the second relations of (3.3.4) between $\frac{\Delta t}{2}$ and Δt.
- It is no more necessary to solve (using auxiliary variables) between 0 and $-\frac{\Delta t}{2}$, and between 0 and $\frac{\Delta t}{2}$ (as its is done in [Diaz 2009][Grote 2009]).
- Taking into account the right-hand side terms associated with the absorbing boundary condition in (1.2.16), as well as source terms characterizing conductive materials (with a current density computed using Ohm's law), should be straightforward in the previous scheme.

3.3.2 Algorithmic aspects

We now discuss implementation aspects of the three steps scheme (3.3.9) more precisely, how to advance the electromagnetic field components from $t^n = t^0 + n\Delta t$ to $t^{n+1} = t^0 + (n+1)\Delta t$. Let Δt_c and Δt_F respectively define the minimum time steps of the coarse and fine grid parts from which we deduce the splitting of $\mathbb{E}$ and $\mathbb{H}$. We define $p = \dfrac{\Delta t_c}{\Delta t_F}$ (we assume that p is an integer, and we shall make a distinction in practice between odd and even values of p). The reference time step is $\Delta t = \Delta t_c$. In the fine part of the grid, we iterate from t^n to $t^n + \Delta t$ using p iterations with time step $\Delta t_F : t^n + \tau$ with $\tau \in [0\ ,\ \Delta t]$. In the three step scheme, the electric field $\mathbb{E}(t)$ and the magnetic field $\mathbb{H}(t)$ are known quantities at the time station t^n. We introduce the following intermediate quantities that are needed in the implementation of the three steps scheme (3.3.9) :

$$
\left\{
\begin{array}{lcll}
\mathbb{U}^n & = & \mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n) & = \mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n) \\
\mathbb{V}^{n+\frac{1}{2}} & \approx & \mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\mathbb{E}(t^n+\dfrac{\Delta t}{2}) & = \mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\left(\mathbb{E}(t^n)+\dfrac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}(t^n)\right) \\
 & & & = \mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\left(\mathbb{E}(t^n)+\dfrac{\Delta t}{2}\mathbb{U}^n+\dfrac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(t^n)\right), \\
\mathbb{W}^{n+1} & \approx & \mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n+\Delta t) & = \mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\left[\mathbb{H}(t^n)+\Delta t\mathbb{S}^\mu\mathbb{E}(t^n+\dfrac{\Delta t}{2})\right] \\
 & & & = \mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n) \\
 & & & + \Delta t\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\left[\mathbb{S}^\mu\mathbb{E}^{[\text{coarse}]}(t^n+\dfrac{\Delta t}{2})+\right. \\
 & & & \qquad \left.\mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(t^n+\dfrac{\Delta t}{2})\right] \\
 & & & = \mathbb{U}^n+\Delta t\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\left[\mathbb{V}^{n+\frac{1}{2}}+\mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(t^n+\dfrac{\Delta t}{2})\right].
\end{array}
\right. \tag{3.3.11}
$$

Note that the variable $\mathbb{W}^{n+1}$ must be computed after the calculation of $\mathbb{E}(t^n + \dfrac{\Delta t}{2})$. Similarly, for the alternative scheme (3.3.10), we can introduce :

$$
\left\{
\begin{array}{lcll}
\mathbb{U}^n & = & \mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n) & = \mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n) \\
\mathbb{V}^{n+\frac{1}{2}} & \approx & \mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\mathbb{E}(t^n+\dfrac{\Delta t}{2}) & = \mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\left(\mathbb{E}(t^n)+\dfrac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}(t^n)\right) \\
 & & & = \mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\left(\mathbb{E}(t^n)+\dfrac{\Delta t}{2}\mathbb{U}^n+\dfrac{\Delta t}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(t^n)\right), \\
\mathbb{W}^{n+1} & \approx & \mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n+\Delta t) & = \mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n)+\Delta t\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{S}^\mu\mathbb{E}(t^n+\dfrac{\Delta t}{2}) \\
 & & & = \mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n)+\Delta t\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{S}^\mu\mathbb{E}(t^n+\dfrac{\Delta t}{2}) \\
 & & & = \mathbb{U}^n+\Delta t\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{S}^\mu\mathbb{E}(t^n+\dfrac{\Delta t}{2}).
\end{array}
\right. \tag{3.3.12}
$$

Using these intermediate variables, the three steps scheme (3.3.9) becomes :

$$\left\{\begin{array}{rcll} \dfrac{d\mathbb{E}}{d\tau} &=& \mathbb{S}^\epsilon\left[(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n)+\mathbb{P}\mathbb{H}(t^n+\tau)\right] & \\ &=& \mathbb{U}^n+\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(t^n+\tau), & \tau\in[0\ ,\ \dfrac{\Delta t}{2}], \\ \dfrac{d\mathbb{H}}{d\tau} &=& \mathbb{V}^{n+\frac12}+\mathbb{S}^\mu\mathbb{P}\mathbb{E}(t^n+\tau), & \\ &=& \mathbb{V}^{n+\frac12}+\mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(t^n+\tau), & \tau\in[0\ ,\ \Delta t], \\ \dfrac{d\mathbb{E}}{d\tau} &=& \mathbb{W}^{n+1}+\mathbb{S}^\epsilon\mathbb{P}\mathbb{H}(t^n+\tau), & \\ &=& \mathbb{W}^{n+1}+\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(t^n+\tau), & \tau\in[\dfrac{\Delta t}{2}\ ,\ \Delta t]. \end{array}\right. \tag{3.3.13}$$

The principle of the local time stepping algorithm is that the leap-frog iteration acts on the whole $\mathbb{E}$ and $\mathbb{H}$ vectors but the degrees of freedom associated with the fine part of the grid require more computations as described below. Let us start by considering the first $\Delta\tau\equiv\Delta t_F$ step :

$$\left\{\begin{array}{rcl} \mathbb{E}(t^n+\dfrac{\Delta\tau}{2}) &=& \mathbb{E}(t^n)+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n)+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{P}\mathbb{H}(t^n) \\ &=& \mathbb{E}(t^n)+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n)+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(0) \\ &=& \mathbb{E}(t^n)+\dfrac{\Delta\tau}{2}\mathbb{U}^n+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(0), \\ \mathbb{H}(t^n+\Delta\tau) &=& \mathbb{H}(t^n)+\Delta\tau\mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\mathbb{E}(t^n+\dfrac{\Delta t}{2})+\Delta\tau\mathbb{S}^\mu\mathbb{P}\mathbb{E}(t^n+\dfrac{\Delta\tau}{2}) \\ &=& \mathbb{H}(t^n)+\Delta\tau\mathbb{S}^\mu\mathbb{E}^{[\text{coarse}]}(t^n+\dfrac{\Delta t}{2})+\Delta\tau\mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(\dfrac{\Delta\tau}{2}) \\ &=& \mathbb{H}(t^n)+\Delta\tau\mathbb{V}^{n+\frac12}+\Delta\tau\mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(\dfrac{\Delta\tau}{2}), \\ \mathbb{E}(t^n+\Delta\tau) &=& \mathbb{E}(t^n+\dfrac{\Delta\tau}{2})+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}(t^n+\Delta t)+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{P}\mathbb{H}(t^n+\Delta\tau) \\ &=& \mathbb{E}(t^n+\dfrac{\Delta\tau}{2})+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n+\Delta t)+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(\Delta\tau) \\ &=& \mathbb{E}(t^n+\dfrac{\Delta\tau}{2})+\dfrac{\Delta\tau}{2}\mathbb{W}^{n+1}+\dfrac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(\Delta\tau), \end{array}\right. \tag{3.3.14}$$

with the following remarks :

- before entering the first substep of (3.3.14), $\mathbb{E}(t^n)$ and $\mathbb{H}(t^n)$ are supposed to be known and stored on the whole grid ;
- the computation of $\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n)$ involves local element-wise operations (with matrices M_i^ϵ and K_i of equation (3.2.6)) for all the elements of the coarse part of the grid, and flux evaluations (with matrix S_{ik} of equation (3.2.6)) at the interfaces between neighboring elements localized in the coarse part of the grid, but also at *hybrid* interfaces (*i.e.* interfaces between two neighboring elements such that one element belongs to the coarse part of the grid and the other one to the fine part of the grid). A similar remark applies to the computation of $\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(0)$ for the elements of the fine part of the grid. $\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n)$ needs to be stored for the subsequent leap-frog iterations ;
- $\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n)$ and $\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(0)$ are also used for the evaluation of $\mathbb{E}(t^n+\dfrac{\Delta t}{2})$. Then, $\mathbb{H}(t^n+\Delta t)$ can be computed (for that, $\mathbb{S}^\mu\mathbb{E}(t^n)$ and $\mathbb{S}^\mu\mathbb{E}(t^n+\dfrac{\Delta t}{2})$ need to be computed, see equation (3.3.11)). Note that the first relation of (3.3.14) and the one defining $\mathbb{V}(t^n+\dfrac{\Delta t}{2})$ are very similar ;

– for the second and third substeps of (3.3.14), $\mathbb{E}(t^n + \frac{\Delta\tau}{2})$ needs to be stored on the whole grid ;
– at the end of the three substeps, $\mathbb{E}(t^n + \Delta\tau)$ and $\mathbb{H}(t^n + \Delta\tau)$ respectively overwrite ;
– finally, for the subsequent leap-frog iterations on the fine grid components, it is worth to store $\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n)$, $\mathbb{S}^\mu\mathbb{E}^{[\text{coarse}]}(t^n + \frac{\Delta t}{2})$ and $\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n + \Delta t)$.

Let us now consider the next $\Delta\tau \equiv \Delta t_F$ step. We have :

$$\left\{\begin{array}{rcl}
\mathbb{E}(t^n + \frac{3\Delta\tau}{2}) &=& \mathbb{E}(t^n + \Delta\tau) + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon(\mathbb{I} - \mathbb{P})\mathbb{H}(t^n) + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{P}\mathbb{H}(t^n + \Delta\tau) \\
&=& \mathbb{E}(t^n + \Delta\tau) + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n) + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(\Delta\tau) \\
&=& \mathbb{E}(t^n + \Delta\tau) + \frac{\Delta\tau}{2}\mathbb{U}^n + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(\Delta\tau), \\
\mathbb{H}(t^n + 2\Delta\tau) &=& \mathbb{H}(t^n + \Delta\tau) + \Delta\tau\mathbb{S}^\mu(\mathbb{I} - \mathbb{P})\mathbb{U}(t^n + \frac{\Delta t}{2}) + \Delta\tau\mathbb{S}^\mu\mathbb{P}\mathbb{E}(t^n + \frac{3\Delta\tau}{2}) \\
&=& \mathbb{H}(t^n + \Delta\tau) + \Delta\tau\mathbb{S}^\mu\mathbb{E}^{[\text{coarse}]}(t^n + \frac{\Delta t}{2}) + \Delta\tau\mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(\frac{3\Delta\tau}{2}) \\
&=& \mathbb{H}(t^n + \Delta\tau) + \Delta\mathbb{V}^{n+\frac{1}{2}} + \Delta\tau\mathbb{S}^\mu\mathbb{E}^{[\text{fine}]}(\frac{3\Delta\tau}{2}), \\
\mathbb{E}(t^n + 2\Delta\tau) &=& \mathbb{E}(t^n + \frac{3\Delta\tau}{2}) + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon(\mathbb{I} - \mathbb{P})\mathbb{V}(t^n + \Delta t) + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{P}\mathbb{H}(t^n + 2\Delta\tau) \\
&=& \mathbb{E}(t^n + \frac{3\Delta\tau}{2}) + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n + \Delta t) + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(2\Delta\tau) \\
&=& \mathbb{E}(t^n + \frac{3\Delta\tau}{2}) + \frac{\Delta\tau}{2}\mathbb{W}^{n+1} + \frac{\Delta\tau}{2}\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(2\Delta\tau)
\end{array}\right. \tag{3.3.15}$$

Note that although in the three substeps of (3.3.15) the whole $\mathbb{E}$ and $\mathbb{H}$ vectors are affected, the computational complexity for updating the coarse and fine parts of these vectors is not the same. For instance, in the first substep, the coarse part of $\mathbb{E}(t^n + \frac{3\Delta\tau}{2})$ is updated from the coarse part of $\mathbb{E}(t^n + \Delta\tau)$ and from $\mathbb{S}^\epsilon\mathbb{H}^{[\text{coarse}]}(t^n)$ (which has been precomputed and stored), while the fine part of $\mathbb{E}(t^n + \frac{3\Delta\tau}{2})$ is updated from the fine part of $\mathbb{E}(t^n + \Delta\tau)$ and from $\mathbb{S}^\epsilon\mathbb{H}^{[\text{fine}]}(\Delta\tau)$ which involves element-wise operations as well as flux evaluations (including at hybrid interfaces).

The case $p = 1$. In this case $\Delta\tau = \Delta t$. For the scheme (3.3.11) we have the following steps :

1. Set $\mathbb{U} = \mathbb{S}^\epsilon(\mathbb{I} - \mathbb{P})\mathbb{H}^n$, $\tilde{\mathbb{U}} = \mathbb{S}^\epsilon\mathbb{P}\mathbb{H}^n$ and $\mathbb{V} = \mathbb{S}^\mu(\mathbb{I} - \mathbb{P})\left(\mathbb{E}^n + \frac{\Delta t}{2}\left[\mathbb{U} + \tilde{\mathbb{U}}\right]\right)$
2. Compute
$$\mathbb{E}^{n+\frac{1}{2}} = \mathbb{E}^n + \frac{\Delta\tau}{2}\left[\mathbb{U} + \tilde{\mathbb{U}}\right]$$
3. Set $\tilde{\mathbb{V}} = \mathbb{S}^\mu\mathbb{P}\mathbb{E}^{n+\frac{1}{2}}$
4. Compute
$$\mathbb{H}^{n+1} = \mathbb{H}^n + \Delta\tau\left[\mathbb{V} + \tilde{\mathbb{V}}\right]$$
5. Set $\mathbb{W} = \mathbb{U} + \Delta t\mathbb{S}^\epsilon(\mathbb{I} - \mathbb{P})\left(\mathbb{V} + \tilde{\mathbb{V}}\right)$
6. Compute
$$\mathbb{E}^{n+1} = \mathbb{E}^{n+\frac{1}{2}} + \frac{\Delta\tau}{2}\left[\mathbb{W} + \mathbb{S}^\epsilon\mathbb{P}\mathbb{H}^{n+1}\right]$$

We can check that this scheme is equivalent to the classical Störmer-Verlet scheme. Moreover, we have two multiplications by $\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})$ (steps 1 and 5), two multiplications by $\mathbb{S}^\epsilon\mathbb{P}$ (steps 1 and 6), one multiplication by $\mathbb{S}^\mu(\mathbb{I}-\mathbb{P})$ (step 1) and one multiplication by $\mathbb{S}^\mu\mathbb{P}$, so that the computational cost is equivalent to the classical Störmer-Verlet scheme (two multiplications by $\mathbb{S}^\epsilon$ and one multiplication by $\mathbb{S}^\mu$).

For the scheme (3.3.12) we have the following steps :

1. Set $\mathbb{U} = \mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}^n$, $\tilde{\mathbb{U}} = \mathbb{S}^\epsilon\mathbb{P}\mathbb{H}^n$ and $\mathbb{V} = \mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\left(\mathbb{E}^n + \dfrac{\Delta t}{2}\left[\mathbb{U}+\tilde{\mathbb{U}}\right]\right)$
2. Compute
$$\mathbb{E}^{n+\frac{1}{2}} = \mathbb{E}^n + \frac{\Delta\tau}{2}\left[\mathbb{U}+\tilde{\mathbb{U}}\right]$$
3. Set $\tilde{\mathbb{V}} = \mathbb{S}^\mu\mathbb{P}\mathbb{E}^{n+\frac{1}{2}}$
4. Compute
$$\mathbb{H}^{n+1} = \mathbb{H}^n + \Delta\tau\left[\mathbb{V}+\tilde{\mathbb{V}}\right]$$
5. Set $\mathbb{W} = \mathbb{U} + \Delta t\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\left(\mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\mathbb{E}^{n+\frac{1}{2}} + \tilde{\mathbb{V}}\right)$
6. Compute
$$\mathbb{E}^{n+1} = \mathbb{E}^{n+\frac{1}{2}} + \frac{\Delta\tau}{2}\left[\mathbb{W} + \mathbb{S}^\epsilon\mathbb{P}\mathbb{H}^{n+1}\right]$$

In this case, we have two multiplications by $\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})$ (steps 1 and 5), two multiplications by $\mathbb{S}^\epsilon\mathbb{P}$ (steps 1 and 6), two multiplications by $\mathbb{S}^\mu(\mathbb{I}-\mathbb{P})$ (steps 1 and 5) and one multiplication by $\mathbb{S}^\mu\mathbb{P}$, so that the computational cost is greater than the classical Störmer-Verlet scheme (two multiplications by $\mathbb{S}^\epsilon$ and one multiplication by $\mathbb{S}^\mu$).

The case $p = 2$. In this case, we set $\Delta\tau = \dfrac{\Delta t}{2}$. For the scheme (3.3.11) we have the following steps [1] :

1. Set $\mathbb{U} = \mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\mathbb{H}^n$, $\tilde{\mathbb{U}} = \mathbb{S}^\epsilon\mathbb{P}\mathbb{H}^n$ and $\mathbb{V} = \mathbb{S}^\mu(\mathbb{I}-\mathbb{P})\left(\mathbb{E}^n + \dfrac{\Delta t}{2}\left[\mathbb{U}+\tilde{\mathbb{U}}\right]\right)$
2. Compute
$$\begin{aligned}
\mathbb{E}^{n+\frac{1}{4}} &= \mathbb{E}^n + \frac{\Delta\tau}{2}\left[\mathbb{U}+\tilde{\mathbb{U}}\right],\\
\mathbb{H}^{n+\frac{1}{2}} &= \mathbb{H}^n + \Delta\tau\left[\mathbb{V} + \mathbb{S}^\mu\mathbb{P}\mathbb{E}^{n+\frac{1}{4}}\right],\\
\mathbb{E}^{n+\frac{1}{2}} &= \mathbb{E}^{n+\frac{1}{4}} + \frac{\Delta\tau}{2}\left[\mathbb{U} + \mathbb{S}^\epsilon\mathbb{P}\mathbb{H}^{n+\frac{1}{2}}\right].
\end{aligned}$$
3. Set $\tilde{\mathbb{V}} = \mathbb{S}^\mu\mathbb{P}\mathbb{E}^{n+\frac{1}{2}}$ and $\mathbb{W} = \mathbb{U} + \Delta t\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})\left(\mathbb{V}+\tilde{\mathbb{V}}\right)$
4. Compute
$$\begin{aligned}
\mathbb{E}^{n+\frac{3}{4}} &= \mathbb{E}^{n+\frac{1}{2}} + \frac{\Delta\tau}{2}\left[\mathbb{W} + \mathbb{S}^\epsilon\mathbb{P}\mathbb{H}^{n+\frac{1}{2}}\right],\\
\mathbb{H}^{n+1} &= \mathbb{H}^{n+\frac{1}{2}} + \Delta\tau\left[\mathbb{V} + \mathbb{S}^\mu\mathbb{P}\mathbb{E}^{n+\frac{3}{4}}\right],\\
\mathbb{E}^{n+1} &= \mathbb{E}^{n+\frac{3}{4}} + \frac{\Delta\tau}{2}\left[\mathbb{W} + \mathbb{S}^\epsilon\mathbb{P}\mathbb{H}^{n+1}\right].
\end{aligned}$$

We have two multiplications by $\mathbb{S}^\epsilon(\mathbb{I}-\mathbb{P})$ (steps 1 and 2), four multiplications by $\mathbb{S}^\epsilon\mathbb{P}$ (steps 1, 2,and 4), one multiplication by $\mathbb{S}^\mu(\mathbb{I}-\mathbb{P})$ (step 1) and three multiplications by $\mathbb{S}^\mu\mathbb{P}$ (steps 2, 3 and 4).

1. We do not write the steps for the scheme (3.3.12) since the only difference is in the definition of $\mathbb{W}$.

The case $p = 3$. In this case, we set $\Delta\tau = \dfrac{\Delta t}{3}$. For the scheme (3.3.11) we have the following steps :

1. Set $\mathbb{U} = \mathbb{S}^{\epsilon}(\mathbb{I} - \mathbb{P})\mathbb{H}^n$, $\tilde{\mathbb{U}} = \mathbb{S}^{\epsilon}\mathbb{P}\mathbb{H}^n$ and $\mathbb{V} = \mathbb{S}^{\mu}(\mathbb{I} - \mathbb{P})\left(\mathbb{E}^n + \dfrac{\Delta t}{2}\left[\mathbb{U} + \tilde{\mathbb{U}}\right]\right)$
2. Compute

$$\begin{aligned}
\mathbb{E}^{n+\frac{1}{6}} &= \mathbb{E}^n + \frac{\Delta\tau}{2}\left[\mathbb{U} + \tilde{\mathbb{U}}\right], \\
\mathbb{H}^{n+\frac{1}{3}} &= \mathbb{H}^n + \Delta\tau\left[\mathbb{V} + \mathbb{S}^{\mu}\mathbb{P}\mathbb{E}^{n+\frac{1}{6}}\right], \\
\mathbb{E}^{n+\frac{1}{3}} &= \mathbb{E}^{n+\frac{1}{6}} + \frac{\Delta\tau}{2}\left[\mathbb{U} + \mathbb{S}^{\epsilon}\mathbb{P}\mathbb{H}^{n+\frac{1}{3}}\right], \\
\mathbb{E}^{n+\frac{1}{2}} &= \mathbb{E}^{n+\frac{1}{3}} + \frac{\Delta\tau}{2}\left[\mathbb{U} + \mathbb{S}^{\epsilon}\mathbb{P}\mathbb{H}^{n+\frac{1}{3}}\right].
\end{aligned}$$

3. Set $\tilde{\mathbb{V}} = \mathbb{S}^{\mu}\mathbb{P}\mathbb{E}^{n+\frac{1}{2}}$
4. Compute

$$\mathbb{H}^{n+\frac{2}{3}} = \mathbb{H}^{n+\frac{1}{3}} + \Delta\tau\left[\mathbb{V} + \tilde{\mathbb{V}}\right]$$

5. Set $\mathbb{W} = \mathbb{U} + \Delta t\mathbb{S}^{\epsilon}(\mathbb{I} - \mathbb{P})\left(\mathbb{V} + \tilde{\mathbb{V}}\right)$
6. Compute

$$\begin{aligned}
\mathbb{E}^{n+\frac{2}{3}} &= \mathbb{E}^{n+\frac{1}{2}} + \frac{\Delta\tau}{2}\left[\mathbb{W} + \mathbb{S}^{\epsilon}\mathbb{P}\mathbb{H}^{n+\frac{2}{3}}\right], \\
\mathbb{E}^{n+\frac{5}{6}} &= \mathbb{E}^{n+\frac{2}{3}} + \frac{\Delta\tau}{2}\left[\mathbb{W} + \mathbb{S}^{\epsilon}\mathbb{P}\mathbb{H}^{n+\frac{2}{3}}\right], \\
\mathbb{H}^{n+1} &= \mathbb{H}^{n+\frac{2}{3}} + \Delta\tau\left[\mathbb{V} + \mathbb{S}^{\mu}\mathbb{P}\mathbb{E}^{n+\frac{5}{6}}\right], \\
\mathbb{E}^{n+1} &= \mathbb{E}^{n+\frac{5}{6}} + \frac{\Delta\tau}{2}\left[\mathbb{W} + \mathbb{S}^{\epsilon}\mathbb{P}\mathbb{H}^{n+1}\right].
\end{aligned}$$

We have two multiplications by $\mathbb{S}^{\epsilon}(\mathbb{I} - \mathbb{P})$ (steps 1 and 4), six multiplications by $\mathbb{S}^{\epsilon}\mathbb{P}$ (steps 1, 2,and 5), one multiplication by $\mathbb{S}^{\mu}(\mathbb{I} - \mathbb{P})$ (step 1) and three multiplications by $\mathbb{S}^{\mu}\mathbb{P}$ (steps 2, 3 and 5).

3.4 Numerical results in 1D

We now present numerical experiments in 1D to validate the proposed local time stepping DGTD method, and evaluate its accuracy and efficiency for several configurations involving different boundary conditions. We solve the following equations :

$$\begin{cases}
\varepsilon\dfrac{\partial E}{\partial t} - \dfrac{\partial H}{\partial x} = 0, \\
\mu\dfrac{\partial H}{\partial t} - \dfrac{\partial E}{\partial x} = 0.
\end{cases}$$

First, we consider the propagation of an eigenmode in a medium delimited by metallic boundaries, in which the analytical solution of Maxwell's equations is known. For this problem, we assess the conservation of a discrete energy for different rates of local refinement p. We then consider a test problem involving the propagation of a Gaussian pulse in a domain with periodic boundaries on both sides. In the third test problem, we repeat the same experience with absorbing boundaries conditions. Lastly, we examine the propagation of a Gaussian pulse in a one-dimensional locally refined mesh with the presence of a space-varying permittivity.

3.4.1 Eigenmode in a 1D interval with metallic boundaries

We consider the propagation of an eigenmode in $\Omega = [0, 1]$ with metallic boundaries at $x = 0$ and $x = 1$ and with constant material parameters $\varepsilon = \mu = 1$. The initial solution is :

$$E(x, 0) = \sin(\pi x) \quad \text{and} \quad H(x, 0) = 0,$$

and the analytical solution is given by :

$$E(x, t) = \sin(\pi x)\cos(\omega t) \quad \text{and} \quad H(x, t) = \cos(\pi x)\sin(\omega t).$$

The interval $[0, 1]$ is first discretized in 70 equally spaced elements. Then, a zone in the middle of the interval is refined by a factor $p = 2, 4$ or 8 as shown on Fig. 3.1 (5 elements initially in the fine grid). Hence if Δx_f and Δx_c respectively denote the discretization step of the fine and coarse grids, we have $\Delta x_f = \Delta x_c/p$. Let Δt_c and Δt_f respectively define the minimum time steps of the coarse and fine grid parts. We recall that in the simulations, the reference time step is Δt_f =CFL$\times\Delta x_f$ for the global time stepping strategy and Δt_c =CFL$\times\Delta x_c$ for the local time stepping one. So, for every time step Δt_c, we shall take $p \geq 2$ local time-steps of size $\Delta t_f = \Delta t_c/p$ inside the refined region with the second order local time stepping scheme. Note that in the absence of local refinement, i.e. $p = 1$, the (local) time stepping scheme corresponds to the standard leap-frog (LF) method. The same values of the CFL number is used for the global and local time stepping strategy i.e. 0.5 and 0.24 for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods respectively.

We first present results for a simulation time T=4×10^{-8} s which corresponds to the time it takes for the signal to travel 6 times back and forth between the two boundaries, i.e. a distance of 12 meters. On Fig. 3.2 we show the time evolution of the electric field E at $x = 0.5$ for the DGTD-$\mathbb{P}_1$ method with a rate of local refinement $p = 2$. On this figure, the three plots that respectively correspond to the exact solution, the numerical solution based on the global time stepping method and the numerical solution resulting from the local time stepping strategy are almost indistinguishable, even on the plots of the right figure where we zoom on a time window toward the end of the simulation. The time evolution of the L_2 error between the exact and numerical solutions for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods, still with $p = 2$, for the global and local time stepping strategies, are shown on Fig. 3.3. While essentially the same accuracy is observed when the approximation order in space is equal to 1, a different behaviour is obtained for the DGTD-$\mathbb{P}_2$ method. For this second case, it seems that the overall accuracy is dominated by the accuracy of the time stepping scheme and although we can expect second order accuracy in time for both the global and local time stepping strategies, it is also clear that the two time integration schemes are characterized by different coefficients in the truncation error expansion. In Fig. 3.4 and Fig. 3.5, we display the time evolution of the L_2 error between the exact and numerical solutions for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods, with $p = 4$ and $p = 8$. For half the time of the simulation, the overall accuracy of the local time stepping schemes for the DGTD-$\mathbb{P}_2$ method with $p = 4$ and $p = 8$ seems to match the one observed for $p = 2$, however a conspicuous instability phenomenon appears henceforth. This stability issue probably stems from the fact that we kept an optimal time step for the coarse grid i.e. $\Delta t_g = \Delta t_{\text{LF}} = 0.24 \times \Delta x_g$, where Δt_{LF} corresponds to the largest time step used in the standard leap-frog (LF) method in the absence of local refinement ($p = 1$). Even when considering the DGTD-$\mathbb{P}_1$ method for a twice longer physical time, namely T=8×10^{-8} s, we already observe on Fig. 3.6, just with the $p = 2$ configuration, that there is a threshold upon which the time evolution of the L_2 error diverges.

In [Diaz 2009], Diaz and Grote ave shown, via numerical experiments, how a small overlap between the fine and the coarse regions permits the use of the optimal time step Δt_{LF}, dictated by the leap-frog method in the coarse region leading to a stable local time stepping scheme. The impact of this overlapping strategy, which consists in extending slightly the region where local time steps are used into that part of the mesh immediately adjacent to the refined region, can be assessed on Fig. 3.7 where we compare the time evolution of L_2 error for the DGTD-$\mathbb{P}_1$ method with $p = 2$ when we use an overlapping area of either 1 or 2 cells. We note that in all the simulations discussed so far, the fine grid part includes two

cells from the coarse grid part at its boundaries and that a single coarse grid cell at the boundaries of the fine grid part is not sufficient to ensure stability of the local time stepping strategy. In [Diaz 2009], the influence of the overlapping strategy on the stability has only been addressed in the framework of a DG method based on $\mathbb{P}_1$ elements and an interior penalty formulation (IPDG) on one hand, and a continuous finite element formulation on the other hand. Therefore, it would be worth to conduct a similar stability analysis for the DGTD formulation considered here, in particular to assess the robustness of the proposed local time stepping strategy for cavity problems.

In Tab. 3.4.1 we summarize the final L_2 errors and CPU times for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods and for different values of p. Note that values associated with the DGTD-$\mathbb{P}_2$ using a local time stepping scheme with $p = 4$ and $p = 8$ are given only as information purposes but the remainder of the L_2 errors values offers us a glimpse of the expected accuracy of the local time stepping method. But the crux of the matter concerns the CPU times where the gains achieved when selecting the local time stepping scheme lead to foreseeable results, that is a noticeable reduction of the CPU time, especially as the rate of local refinement increases.

Finally, to confirm the second order accuracy in time we plot on Fig. 3.8 the time evolution of the L_2 error for the DGTD-$\mathbb{P}_2$ method with $p = 2$ and CFL=0.24 and 0.12 (which thus corresponds to halving the reference time step for the local time stepping strategy). When CFL=0.12, the final L_2 error is 2.431×10^{-5} which is about 5 times lower than the value obtained for CFL=0.24 thus exceeding the ratio expected for a second order convergence rate.

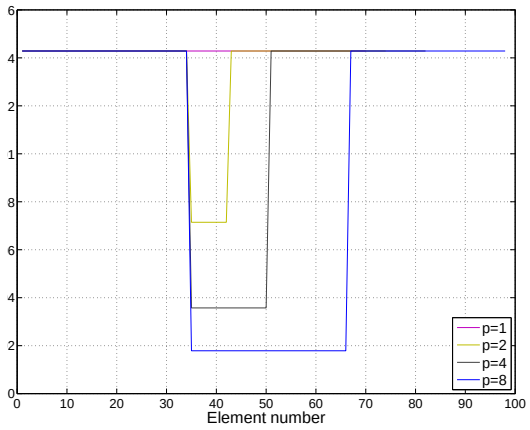


Figure 3.1 – Distribution of points in a 1D interval with metallic boundaries for $p = 2$, $p = 4$ and $p = 8$.

Table 3.4.1 – Eigenmode in a 1D interval with metallic boundaries. Maximum L_2 error in time (simulation time T=4×10^{-8} s).

p	Time stepping strategy	L_2 error DGTD-$\mathbb{P}_1$	CPU (s) DGTD-$\mathbb{P}_1$	L_2 error DGTD-$\mathbb{P}_2$	CPU (s) DGTD-$\mathbb{P}_2$
2	Global Δt	4.467×10^{-3}	0.09	3.222×10^{-5}	0.24
-	Local Δt	4.533×10^{-3}	0.06	1.224×10^{-4}	0.15
4	Global Δt	4.303×10^{-3}	0.19	8.065×10^{-6}	0.49
-	Local Δt	4.391×10^{-3}	0.07	1.612×10^{-3}	0.19
8	Global Δt	4.227×10^{-3}	0.42	2.078×10^{-6}	1.18
-	Local Δt	4.322×10^{-3}	0.13	7.137×10^{-4}	0.33

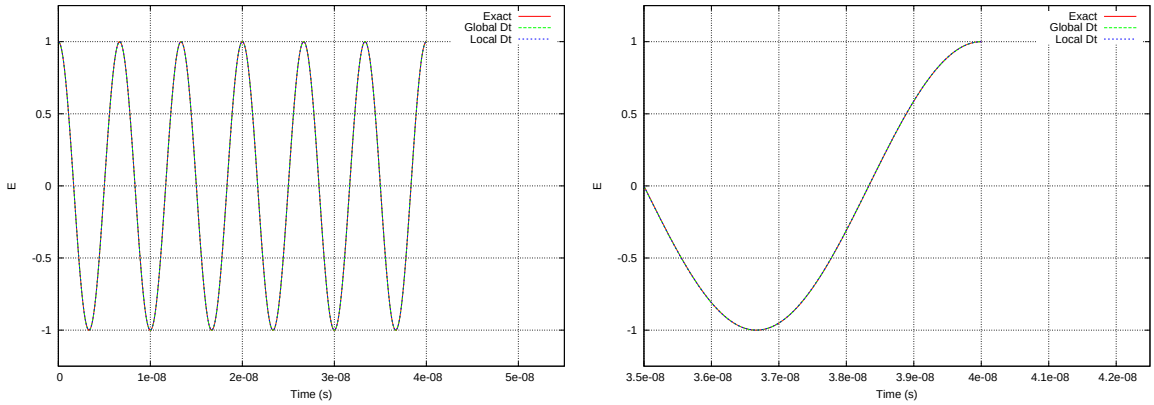
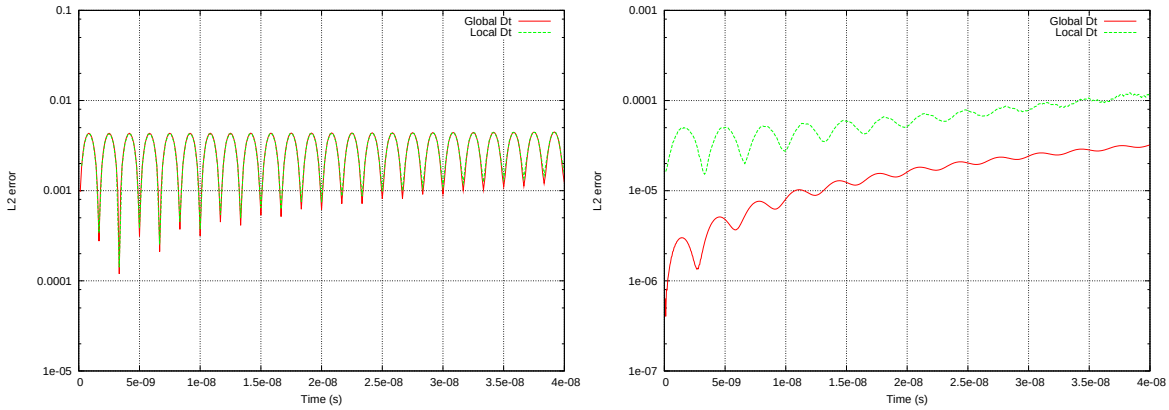
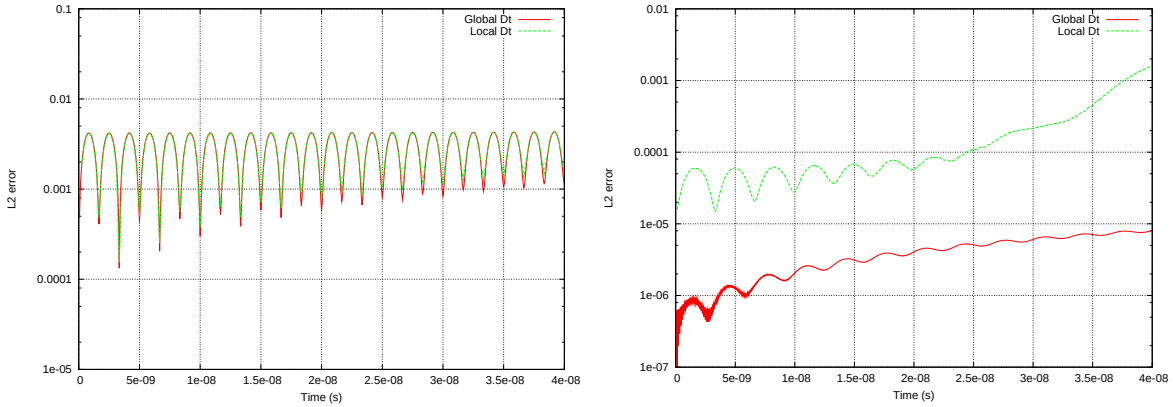


Figure 3.2 – Eigenmode in a 1D interval with metallic boundaries. Time evolution of E at $x = 0.5$ for the DGTD-$\mathbb{P}_1$ method and $p = 2$ (simulation time T=4×10^{-8} s).



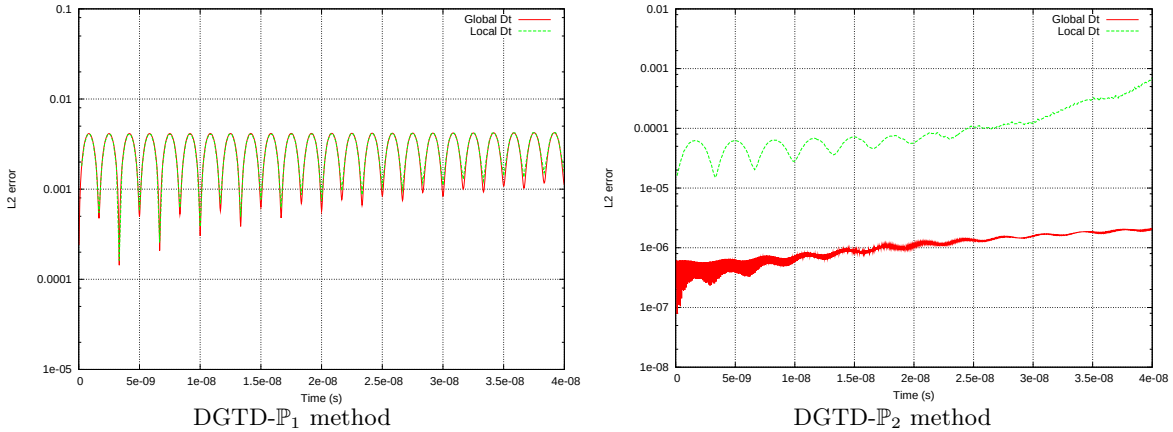
DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.3 – Eigenmode in a 1D interval with metallic boundaries. Time evolution of the L_2 error for $p = 2$ (simulation time T=4×10^{-8} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.4 – Eigenmode in a 1D interval with metallic boundaries. Time evolution of the L_2 error for $p = 4$ (simulation time T=4×10^{-8} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.5 – Eigenmode in a 1D interval with metallic boundaries. Time evolution of the L_2 error for $p = 8$ (simulation time T=4×10^{-8} s).

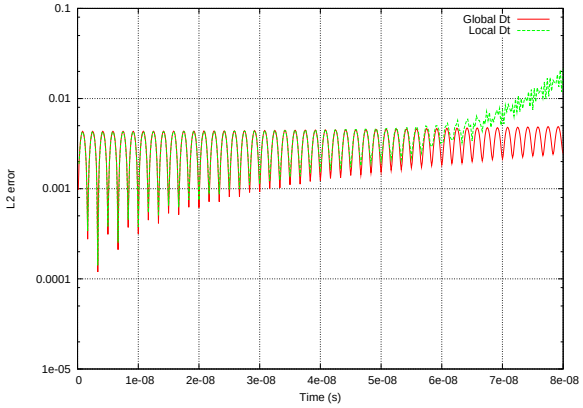


Figure 3.6 – Eigenmode in a 1D interval with metallic boundaries. Time evolution of the L_2 error for the DGTD-$\mathbb{P}_1$ method and $p = 2$ (simulation time T=8×10^{-8} s).

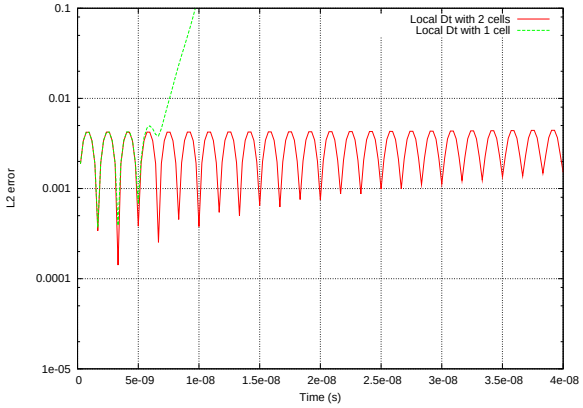


Figure 3.7 – Eigenmode in a 1D interval with metallic boundaries. Time evolution of the L_2 error for the DGTD-$\mathbb{P}_1$ method and $p = 2$ (simulation time T=4×10^{-8} s). Influence of the overlapping strategy.

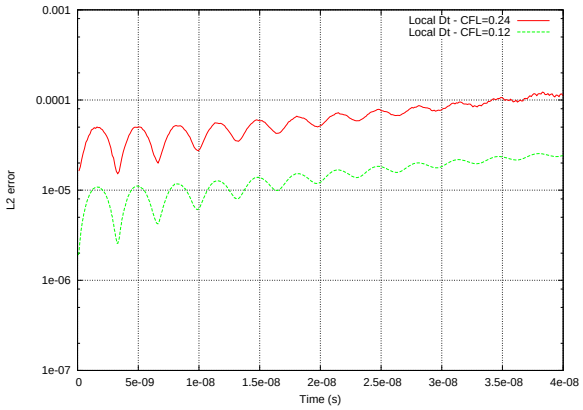


Figure 3.8 – Eigenmode in a 1D interval with metallic boundaries. Time evolution of the L_2 error for the DGTD-$\mathbb{P}_2$ method (simulation time T=4×10^{-8} s). Influence of the reference time step Δt_c.

3.4.2 Gaussian pulse with periodic boundaries

We now consider the propagation of a Gaussian pulse in $\Omega = [0, 6]$ with periodic boundaries at $x = 0$ and $x = 6$, still with constant material parameters $\varepsilon = \mu = 1$. The interval $[0, 6]$ is first discretized in 300 equally spaced elements. Then, similarly to the previous test problem, a zone in the middle of the interval is refined by a factor $p = 2, 4$ or 8 as shown on Fig. 3.9 (50 elements initially in the fine grid). We keep the same notations as previously for the coarse and fine grids parameters and the same values of the CFL number will be used for the global and local time stepping strategy i.e. 0.5 and 0.24 for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods respectively.

The following series of results correspond to a simulation time T=7.5×10^{-8} s i.e. a final distance of 22.5 meters. The center of the Gaussian pulse is initially put at the first quarter of the domain *i.e.* at $x = 1.5$ meters from the left boundary. On Fig. 3.10 and 3.11 we plot the time evolution of the electric field E for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods with a rate of local refinement $p = 2$. On these figures, the three curves respectively correspond to the exact solution, the numerical solution based on the global time stepping strategy and the numerical solution resulting from the local time stepping strategy. These plots are almost indistinguishable on the left figures, but a slight phase error can be observed on the plot of the right figure (zoom on a time window toward the end of the simulation) for the DGTD-$\mathbb{P}_1$ method. However this discrepancy vanishes for a higher interpolation order. The time evolution of the L_2 error between the exact and numerical solutions for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods, for both the global and local time stepping strategies, with $p = 2, 4$ or 8, are shown on Fig. 3.12, 3.13 and 3.14. Comparing to the propagation of the eigenmode in a one-dimensional cavity problem, it straightly appears that first, the accuracy we get when the approximation degree in space is equal to 1 is from now on not exactly the same between the two time stepping strategies, and second, the asymptotic values of the different related curves are approximately 10 times less accurate than the ones observed in the first numerical experiment. Again, for the DGTD-$\mathbb{P}_2$ method, the curve related to the local time stepping scheme shows the same evolution than the one related to the global time stepping scheme with roughly the same difference of accuracy we had in the eigenmode case for $p = 2$. Likewise, this gap between these curves increases while raising the rate of local refinement but simulations presently yield stable results and enforce the assumption that the overall accuracy seems to be dominated by the accuracy of the time stepping scheme.

In Tab. 3.4.2 we summarize the final L_2 errors and CPU times for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods and for different values of p. What is now worth noticing is that the final L_2 error remains close to 2×10^{-3} independently of the values of p. Similarly to the former test problem, Tab. 3.4.2 exhibits significant performance ratios quite promising for 2D and 3D problems.

To conclude, Fig. 3.15 comes to put forward the second order accuracy in time of the local time stepping method. Here again, results closely match expectations.

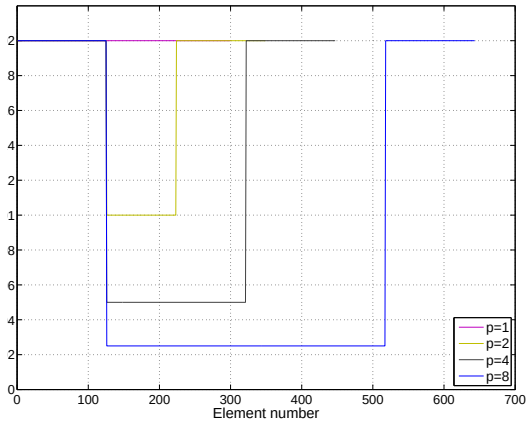


Figure 3.9 – Distribution of points in a 1D interval with periodic boundaries for $p = 2$, $p = 4$ and $p = 8$.

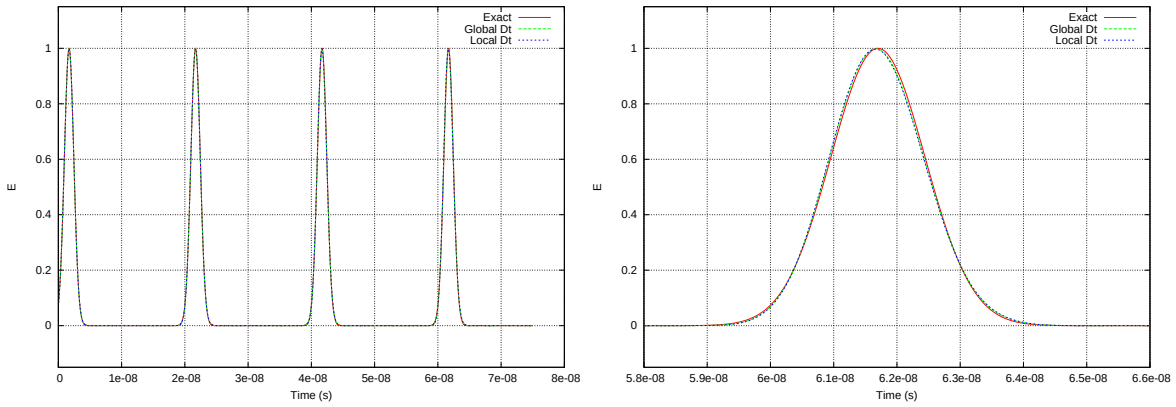


Figure 3.10 – Gaussian pulse with periodic boundaries. Time evolution of E at $x = 2.0$ for the DGTD-$\mathbb{P}_1$ method and $p = 2$ (simulation time T=7.5×10^{-8} s).

3.4.3 Gaussian pulse with absorbing boundaries

Let us now study the case of the propagation of a Gaussian pulse in $\Omega = [0, 6]$ with absorbing boundaries at $x = 0$ and $x = 6$, still with constant material parameters $\varepsilon = \mu = 1$. The interval $[0, 6]$ is first discretized in 300 equally spaced elements. Then, similarly to the previous test problem, a zone in the middle of the interval is refined by a factor $p = 2, 4$ or 8 as shown on Fig. 3.16 (50 elements initially in the fine grid). We keep the same notations as previously for the coarse and fine grids' parameters and the same values of the CFL number will be used for the global and local time stepping strategy i.e. 0.5 and 0.24 for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods respectively. The simulation time has been fixed to T=1.5×10^{-8} s which corresponds to a final distance of 4.5 meters. Here again, the center of the Gaussian pulse is initially put at the first quarter of the domain, that is at 1.5 meters from the left boundary. As previously, we represent on Fig. 3.17 the time evolution of the electric field E for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods with a rate of local refinement $p = 2$. Again, the time evolution of the L_2 error between the exact and numerical solutions for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods, for

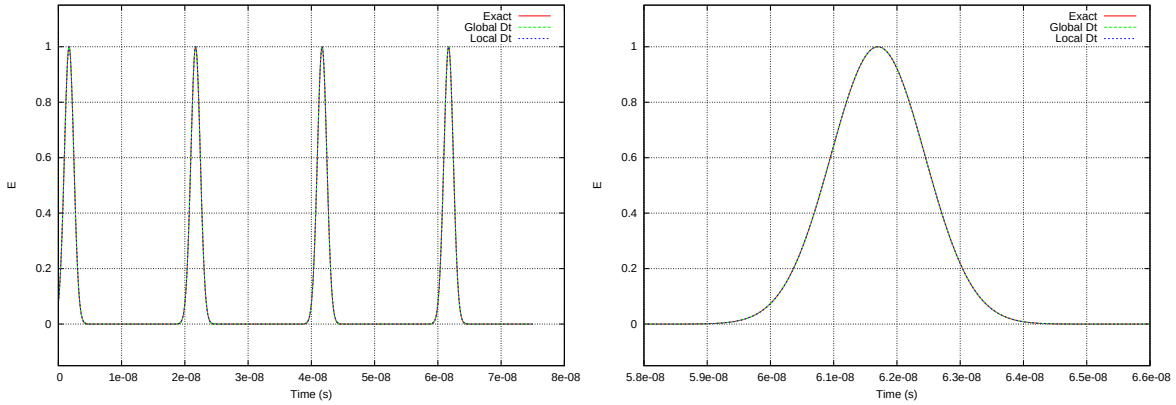
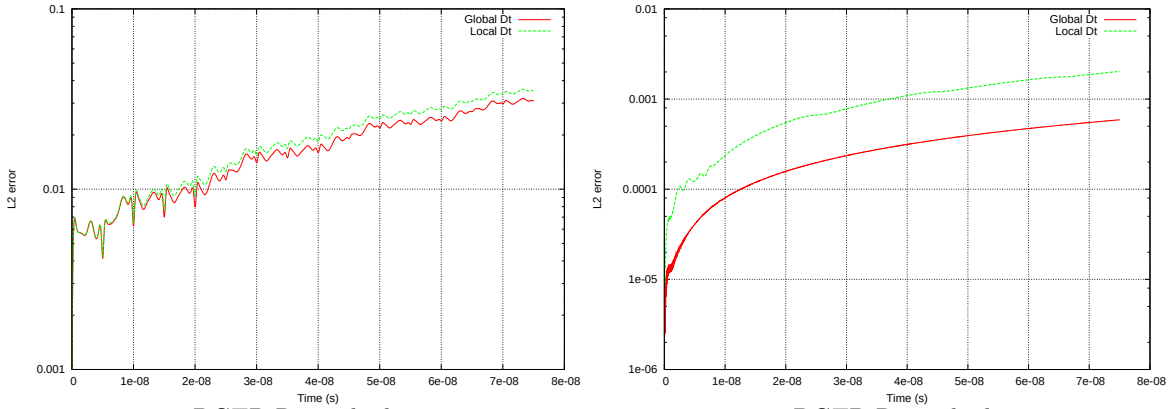
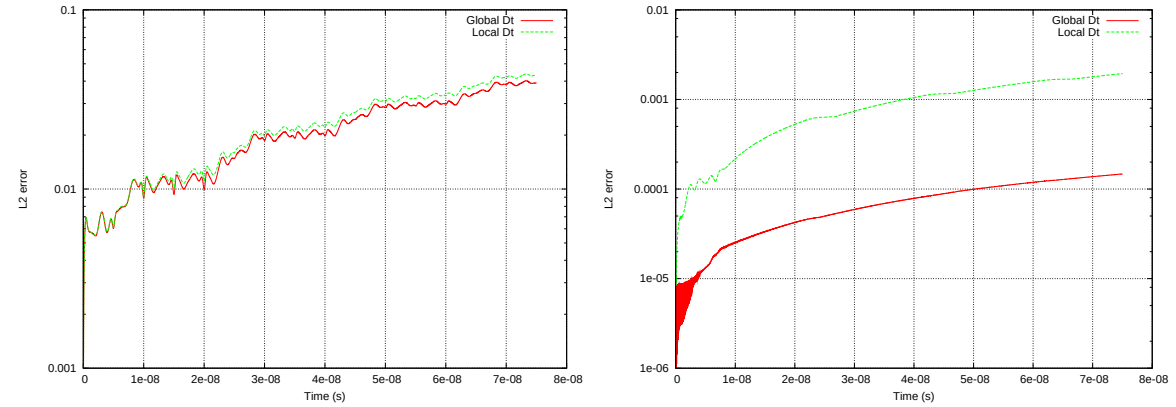


Figure 3.11 – Gaussian pulse with periodic boundaries. Time evolution of E at $x = 2.0$ for the DGTD-$\mathbb{P}_2$ method and $p = 2$ (simulation time T=7.5×10^{-8} s).



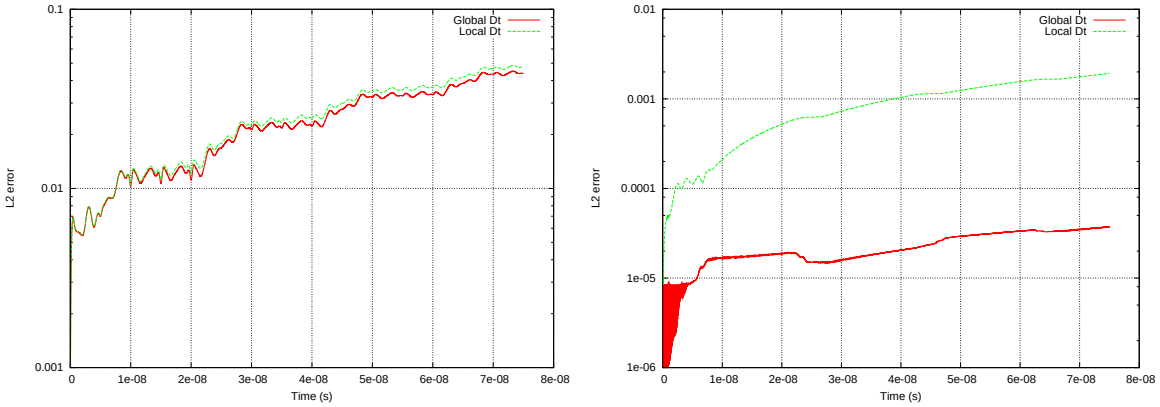
DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.12 – Gaussian pulse with periodic boundaries. Time evolution of the L_2 error for $p = 2$ (simulation time T=7.5×10^{-8} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.13 – Gaussian pulse with periodic boundaries. Time evolution of the L_2 error for $p = 4$ (simulation time T=7.5×10^{-8} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.14 – Gaussian pulse with periodic boundaries. Time evolution of the L_2 error for $p = 8$ (simulation time T=7.5×10^{-8} s).

Table 3.4.2 – Gaussian pulse with periodic boundaries. Maximum L_2 error in time (simulation time T=7.5×10^{-8} s).

p	Time stepping strategy	L_2 error DGTD-$\mathbb{P}_1$	CPU (s) DGTD-$\mathbb{P}_1$	L_2 error DGTD-$\mathbb{P}_2$	CPU (s) DGTD-$\mathbb{P}_2$
2	Global Δt	3.184×10^{-2}	0.47	5.889×10^{-4}	1.35
-	Local Δt	3.587×10^{-2}	0.37	2.027×10^{-3}	1.05
4	Global Δt	4.019×10^{-2}	1.25	1.473×10^{-4}	3.50
-	Local Δt	4.392×10^{-2}	0.57	1.946×10^{-3}	1.72
8	Global Δt	4.517×10^{-2}	3.26	3.738×10^{-5}	10.50
-	Local Δt	4.861×10^{-2}	1.20	1.923×10^{-3}	3.59

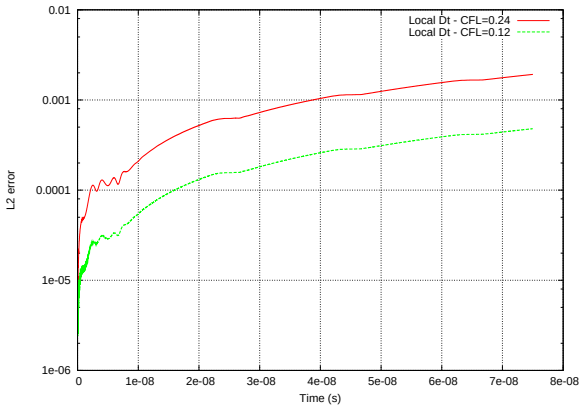


Figure 3.15 – Gaussian pulse with periodic boundaries. Time evolution of the L_2 error for the DGTD-$\mathbb{P}_2$ method (simulation time T=7.5×10^{-8} s). Influence of the reference time step Δt_c.

both the global and local time stepping strategies, still with $p = 2, 4$ or 8, are shown on Fig. 3.19, 3.20 and 3.21. Based on the numerical interpretation we had so far, these results show no overwhelming surprise and curves are plotted in accordance with theoretical limitations. Yet it occurs that the unstable behaviour we could not avoid for cavity problems is no more an issue for open problem settings.

Tab. 3.4.3 gathers the related final L_2 errors and CPU times for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods and for different values of p. Once again, although we limit ourselves to low values of the rate of local refinement, the computational performances of the local time stepping strategy are encouraging in view of its adaptation to 2D and 3D problems.

Fig. 3.22 shows a singular behaviour of the local time stepping method when the pulse approaches the right boundary of the domain as we note that the difference between the two plots seems to shrink. Further investigations should be performed to explain this phenomenon.

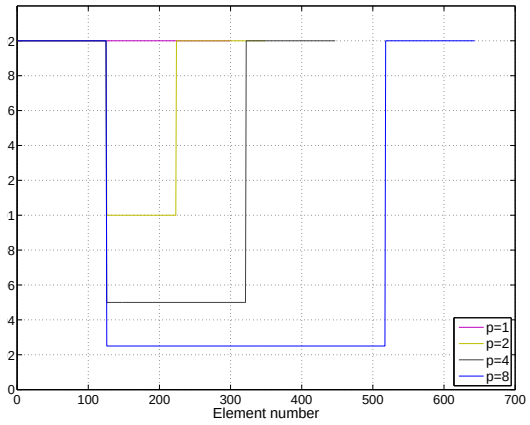
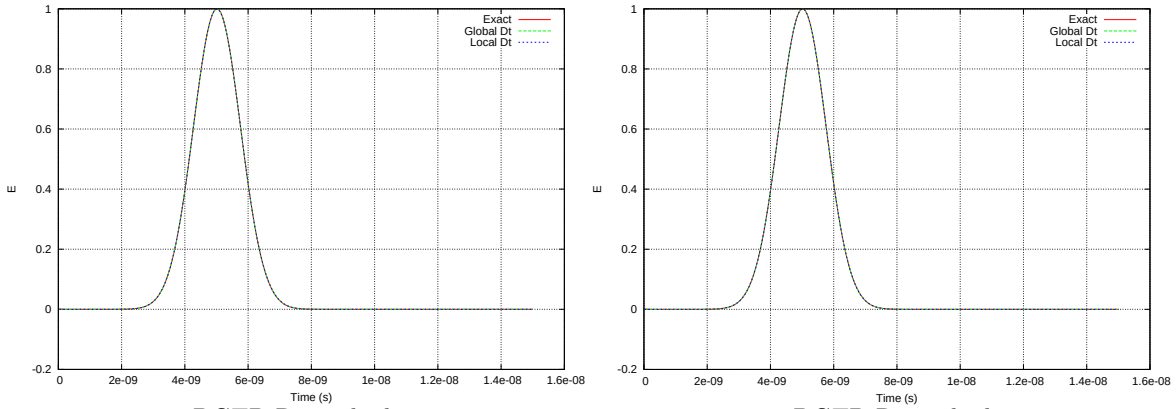


Figure 3.16 – Distribution of points in a 1D interval with absorbing boundaries for $p = 2$, $p = 4$ and $p = 8$.

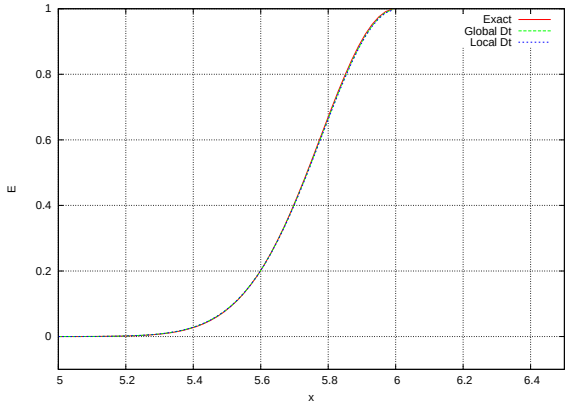


DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

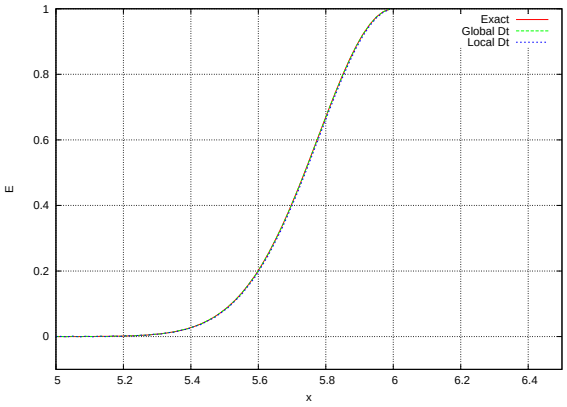
Figure 3.17 – Gaussian pulse with absorbing boundaries. Time evolution of E at $x = 3.0$ for $p = 2$ (simulation time T=1.5×10^{-8} s).

3.4.4 Gaussian pulse with periodic boundaries in a heterogeneous medium

So far, we limited ourselves to propagation problems in homogeneous media. Hence we consider now a configuration featuring the support of heterogeneous material properties. We examine the propagation

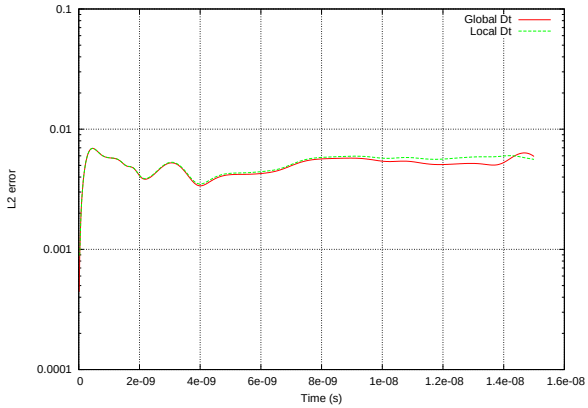


DGTD-$\mathbb{P}_1$ method

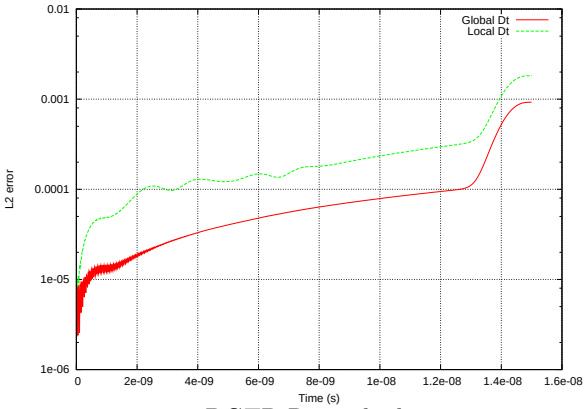


DGTD-$\mathbb{P}_2$ method

Figure 3.18 – Gaussian pulse with absorbing boundaries. x-wise distribution of E for $p = 2$ (simulation time T=1.5×10^{-8} s).

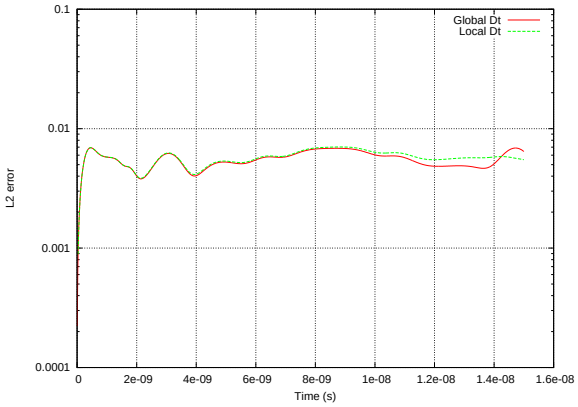


DGTD-$\mathbb{P}_1$ method

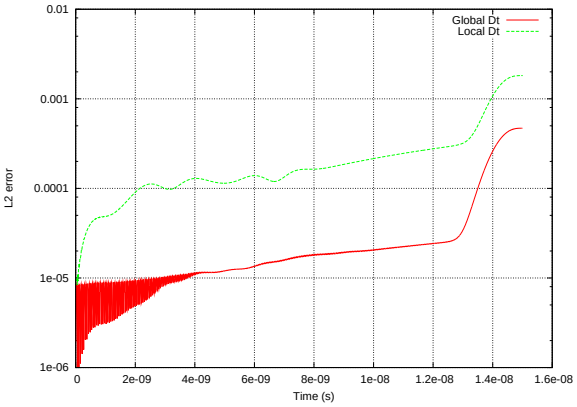


DGTD-$\mathbb{P}_2$ method

Figure 3.19 – Gaussian pulse with absorbing boundaries. Time evolution of the L_2 error for $p = 2$ (simulation time T=1.5×10^{-8} s).

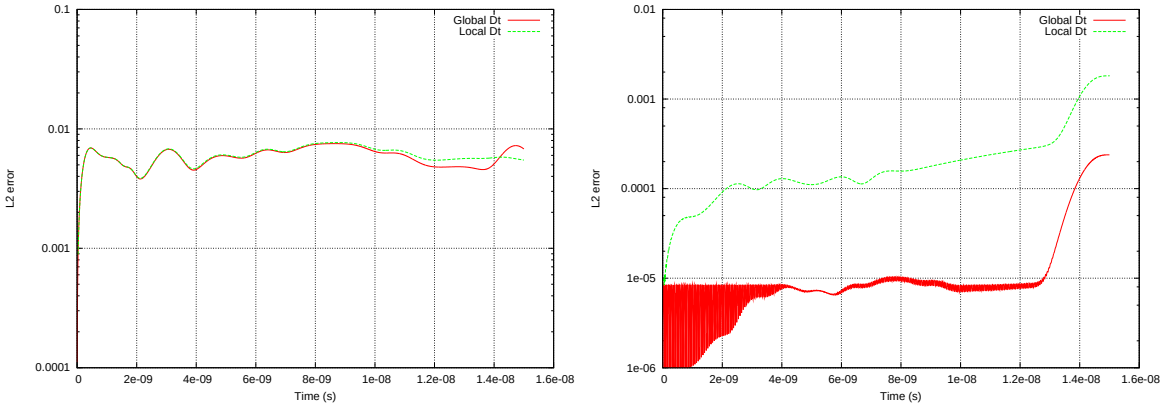


DGTD-$\mathbb{P}_1$ method



DGTD-$\mathbb{P}_2$ method

Figure 3.20 – Gaussian pulse with absorbing boundaries. Time evolution of the L_2 error for $p = 4$ (simulation time T=1.5×10^{-8} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.21 – Gaussian pulse with absorbing boundaries. Time evolution of the L_2 error for $p = 8$ (simulation time T=1.5×10^{-8} s).

Table 3.4.3 – Gaussian pulse with absorbing boundaries. Maximum L_2 error in time (simulation time T=1.5×10^{-8} s).

p	Time stepping strategy	L_2 error DGTD-$\mathbb{P}_1$	CPU (s) DGTD-$\mathbb{P}_1$	L_2 error DGTD-$\mathbb{P}_2$	CPU (s) DGTD-$\mathbb{P}_2$
2	Global Δt	6.920×10^{-3}	0.30	9.245×10^{-4}	0.92
-	Local Δt	6.920×10^{-3}	0.18	1.821×10^{-3}	0.54
4	Global Δt	6.921×10^{-3}	0.76	4.698×10^{-4}	2.35
-	Local Δt	7.024×10^{-3}	0.27	1.819×10^{-3}	0.75
8	Global Δt	7.521×10^{-3}	2.31	2.384×10^{-4}	6.90
-	Local Δt	7.679×10^{-3}	0.44	1.818×10^{-3}	1.40

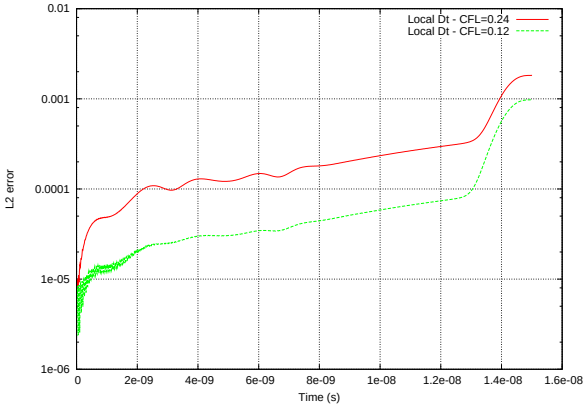


Figure 3.22 – Gaussian pulse with absorbing boundaries. Time evolution of the L_2 error for the DGTD-$\mathbb{P}_2$ method (simulation time T=1.5×10^{-8} s). Influence of the reference time step Δt_c.

of a Gaussian pulse in the domain $\Omega = [0, 6]$ with periodic boundaries at $x = 0$ and $x = 6$. The interval $[0, 6]$ is first discretized in 300 equally spaced elements. Then, similarly to the previous test problems, a region in the middle of the interval is refined by a factor $p = 2, 4$ or 8 as shown on Fig. 3.23 (61 elements initially in the fine grid). On the coarse grid the medium is characterized by the permittivity of vacuum and an electric permittivity equal to 2 is imposed on the remaining fine grid. Fig. 3.24, 3.25 and 3.26 and in Tab. 3.4.4 appears closely similar to those obtained in a homogeneous medium achieving very satisfying results.

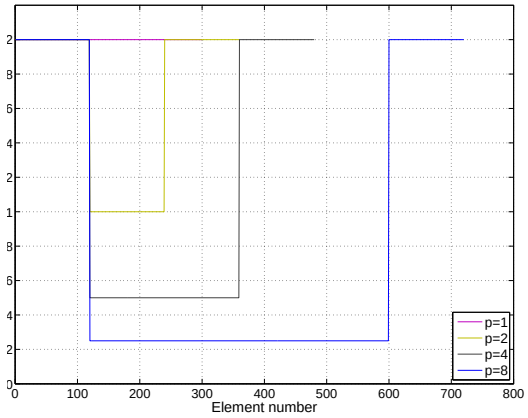
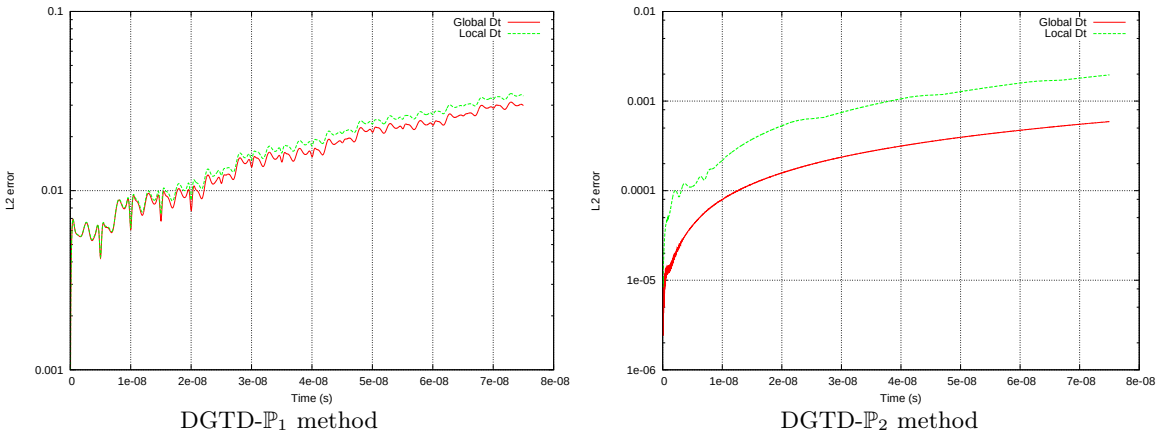
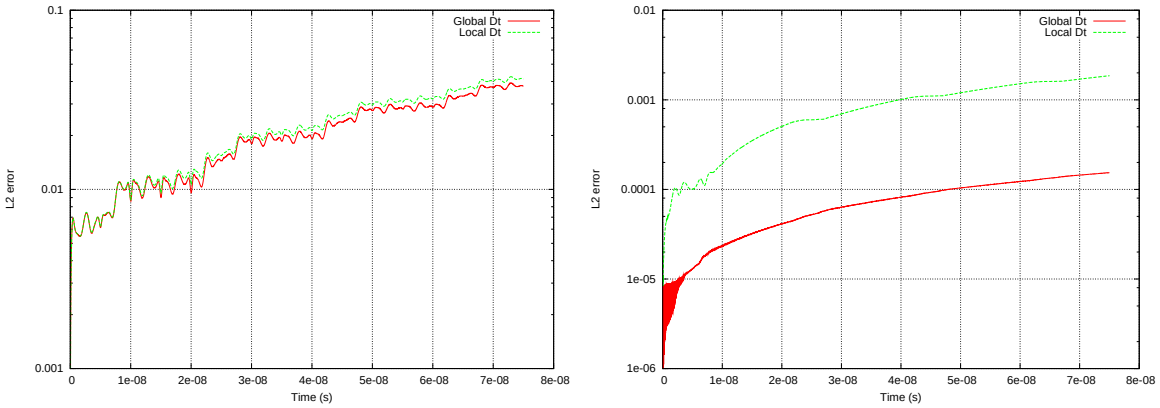


Figure 3.23 – Distribution of points in a 1D interval with periodic boundaries in a heterogeneous medium for $p = 2$, $p = 4$ and $p = 8$.



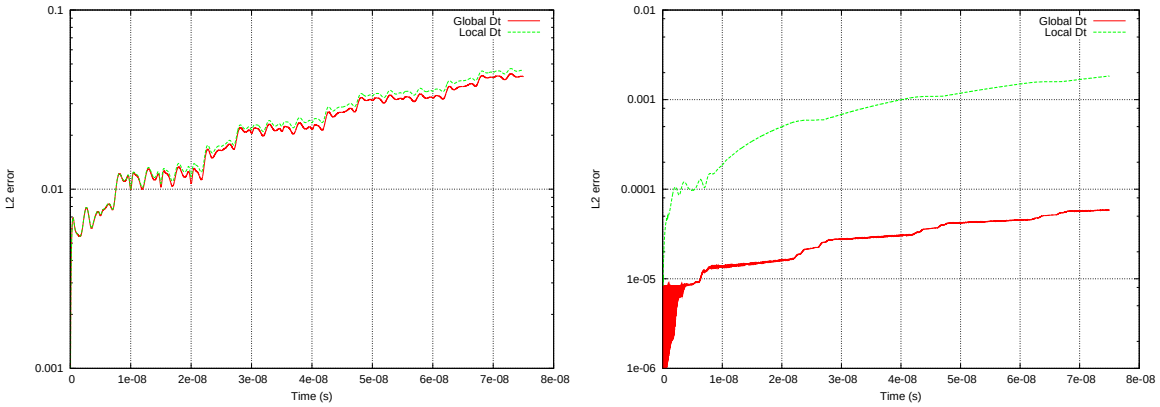
DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.24 – Gaussian pulse with periodic boundaries in a heterogeneous medium. Time evolution of the L_2 error for $p = 2$ (simulation time T=7.5×10^{-8} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.25 – Gaussian pulse with periodic boundaries in a heterogeneous medium. Time evolution of the L_2 error for $p = 4$ (simulation time T=7.5×10^{-8} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.26 – Gaussian pulse with periodic boundaries in a heterogeneous medium. Time evolution of the L_2 error for $p = 8$ (simulation time T=7.5×10^{-8} s).

Table 3.4.4 – Gaussian pulse with periodic boundaries in a heterogeneous medium. Maximum L_2 error in time (simulation time T=7.5×10^{-8} s).

p	Time stepping strategy	L_2 error DGTD-$\mathbb{P}_1$	CPU (s) DGTD-$\mathbb{P}_1$	L_2 error DGTD-$\mathbb{P}_2$	CPU (s) DGTD-$\mathbb{P}_2$
2	Global Δt	3.117×10^{-2}	0.48	5.902×10^{-4}	1.41
-	Local Δt	3.481×10^{-2}	0.37	1.960×10^{-3}	1.17
4	Global Δt	3.925×10^{-2}	1.24	1.539×10^{-4}	3.75
-	Local Δt	4.256×10^{-2}	0.63	1.860×10^{-3}	1.97
8	Global Δt	4.409×10^{-2}	3.62	5.862×10^{-5}	11.72
-	Local Δt	4.711×10^{-2}	1.37	1.835×10^{-3}	4.10

3.5 Numerical results in 2D

We now turn to the numerical solution of the 2D Maxwell equations and consider the case of the transverse magnetic (TM) mode :

$$\begin{cases} \mu \dfrac{\partial H_x}{\partial t} + \dfrac{\partial E_z}{\partial y} = 0, \\ \mu \dfrac{\partial H_y}{\partial t} - \dfrac{\partial E_z}{\partial x} = 0, \\ \varepsilon \dfrac{\partial E_z}{\partial t} - \dfrac{\partial H_y}{\partial x} + \dfrac{\partial H_x}{\partial y} = -J_z. \end{cases}$$

As in the 1D case, we first study the propagation of an eigenmode in a unitary square cavity delimited by metallic boundaries, in which the analytical solution of Maxwell's equations is known. In that case $\varepsilon = \mu = 1$ and $J_z = 0$. We then consider a test problem involving the propagation of a wave initiated by a localized source current. The computational domain is again a unitary square but this time delimited by absorbing boundaries. In that case $\varepsilon = \mu = 1$ and $J_z = \delta(x - x_s, y - y_s) f(t)$.

3.5.1 Eigenmode in square cavity

We consider the propagation of an eigenmode in a unitary perfectly electric conducting square cavity *i.e.* $\Omega = [0,1] \times [0,1]$. The initial solution is :

$$\begin{cases} H_x(x,y,t) &= 0, \\ H_y(x,y,t) &= 0, \\ E_z(x,y,t) &= \sin(\pi x)\sin(\pi y), \end{cases}$$

and the analytical solution is given by :

$$\begin{cases} H_x(x,y,t) &= -\dfrac{\pi}{\omega}\sin(\pi x)\cos(\pi y)\sin(\omega t), \\ H_y(x,y,t) &= \dfrac{\pi}{\omega}\cos(\pi x)\sin(\pi y)\sin(\omega t), \\ E_z(x,y,t) &= \sin(\pi x)\sin(\pi y)\cos(\omega t). \end{cases}$$

For this test problem we make use of an unstructured triangular mesh consisting of 1667 vertices and 3118 triangles (see Fig. 3.27) which is artificially refined in the lower left corner of the square cavity. The ratio between the maximum and minimum values of the local time step is approximately equal to 168. For the local time stepping strategy the partitioning of the mesh is such that the fine and coarse parts respectively consist of 844 and 2274 elements. The corresponding ratio between the minimum time steps of the coarse (Δt_c) and fine (Δt_f) submeshes is approximately equal to 48.4 consequently we have fixed $p = 50$ in the local time stepping strategy and the time step Δt_f is redefined accordingly as $\Delta t_f = \Delta t_c / p$. The objective here is not to assess the performances of the local time stepping strategy but instead to study its accuracy and stability numerically. As for the 1D simulations, the same values of the CFL number is used for the global and local time stepping strategy i.e. 0.33 and 0.16 for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods respectively.

We first present results for a simulation time T=3.3×10^{-8} s. The time evolution of the L_2 error between the exact and numerical solutions for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods, for the global and local time stepping strategies, are shown on Fig. 3.28. As in the 1D case, we observe that after a certain number of periods, the simulations with the local time stepping strategy become unstable. If we now compare the numerical solutions resulting from the global and local time stepping strategies on a shorter time interval *i.e.* in the present case, T=10^{-8} s (see Fig. 3.29), we note that these numerical solutions almost perfectly fit and the discrete energy is almost exactly conserved (see Fig. 3.30).

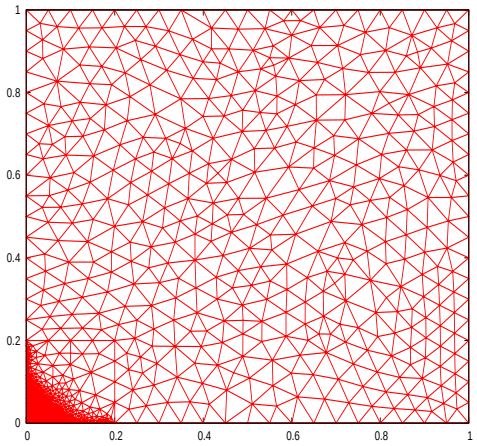


Figure 3.27 – Eigenmode in a square cavity with metallic boundaries. Unstructured triangular mesh.

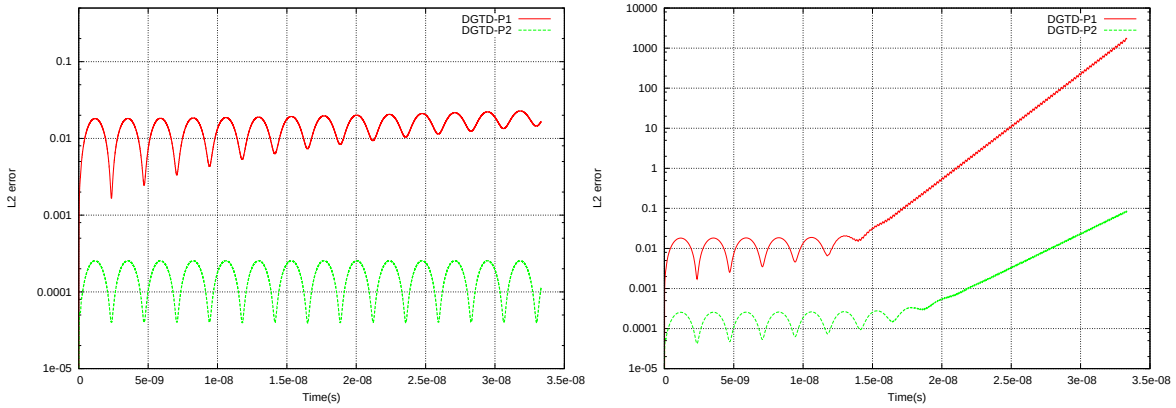
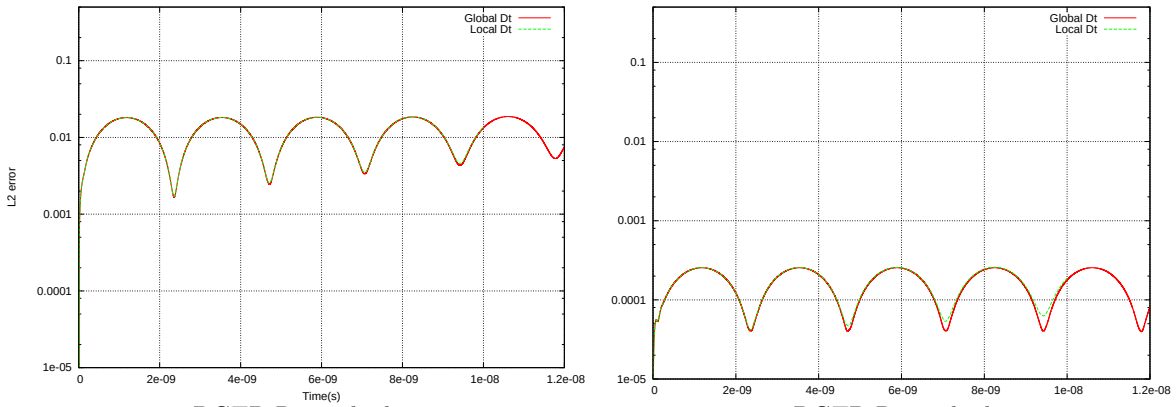
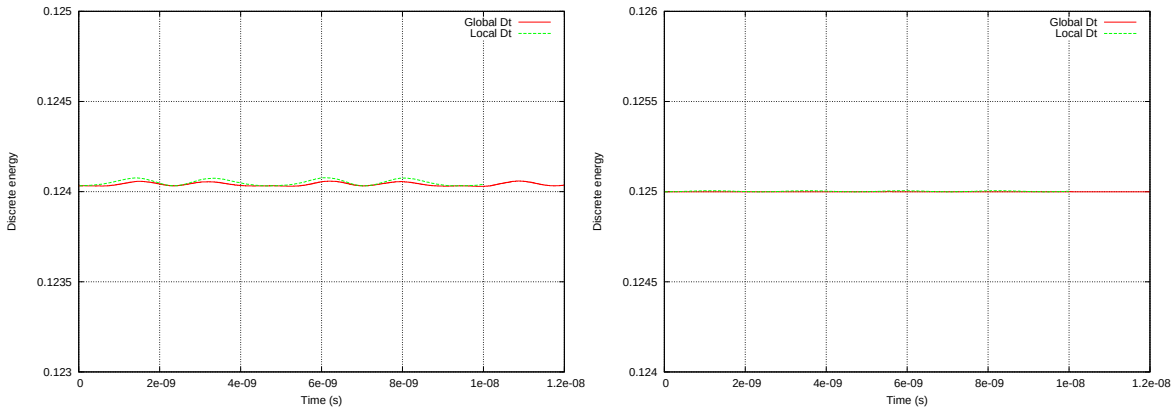


Figure 3.28 – Eigenmode in a square cavity with metallic boundaries. Time evolution of the L_2 error for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods (simulation time T=3.3×10^{-8} s). Global time stepping strategy (left figure) and local ($p = 50$) time stepping strategy (right figure).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.29 – Eigenmode in a square cavity with metallic boundaries. Time evolution of the L_2 error for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods (simulation time T=10^{-8} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.30 – Eigenmode in a square cavity with metallic boundaries. Time evolution of the discrete energy for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods (simulation time T=10^{-8} s).

3.5.2 Wave initiated by a localized source in vacuum

We now consider a problem which is set in a domain with an absorbing boundary condition only. This problem consists in the propagation of an electromagnetic wave emitted by a localized source in a unitary square domain *i.e.* $\Omega = [0,1] \times [0,1]$ with a Silver-Müller absorbing boundary condition set on $\partial\Omega$.

For this test problem we make use of an unstructured triangular mesh consisting of 803 vertices and 1524 triangles (see Fig. 3.31) which is refined in the middle of the domain where the source is localized. The ratio between the maximum and minimum values of the local time step is approximately equal to 45. For the local time stepping strategy the partitioning of the mesh is such that the fine and coarse parts respectively consist of 148 and 1376 elements. The corresponding ratio between the minimum time steps of the coarse (Δt_c) and fine (Δt_f) submeshes is approximately equal to 9.6 consequently we have fixed $p = 10$ in the local time stepping strategy and the time step Δt_f is redefined accordingly as $\Delta t_f = \Delta t_c / p$. The time evolution of the E_z and H_y components at a fixed point of the domain, for the global and local time stepping strategies, are shown on Fig. 3.32 and 3.33. The time evolution of the discrete energy is shown on Fig. 3.34. This time, we do not observe an unstable behavior as it was the case with the test problem considered previously. Besides, the global time step and local time step based simulations with the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods yield numerical solutions that are almost indistinguishable.

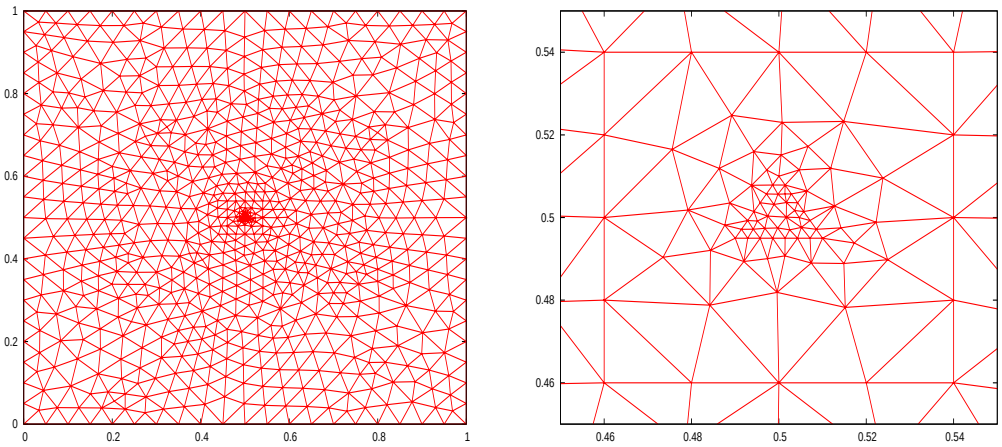
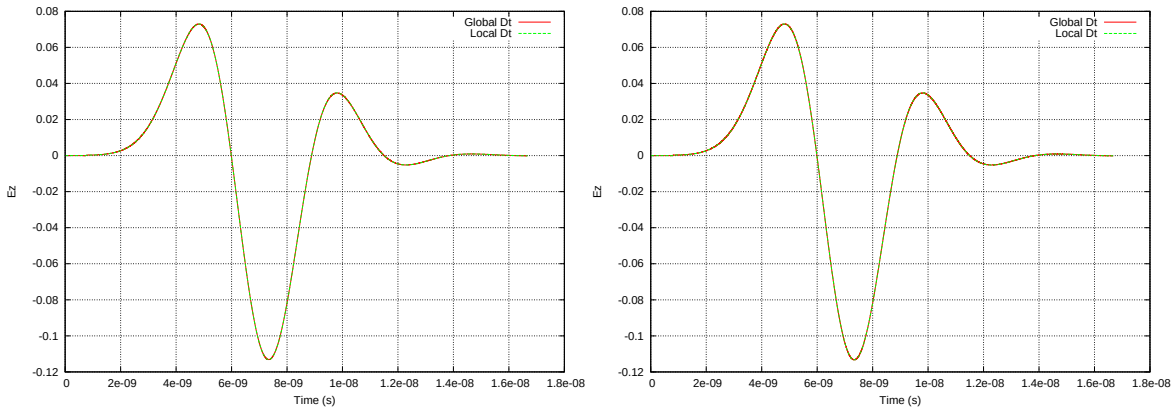
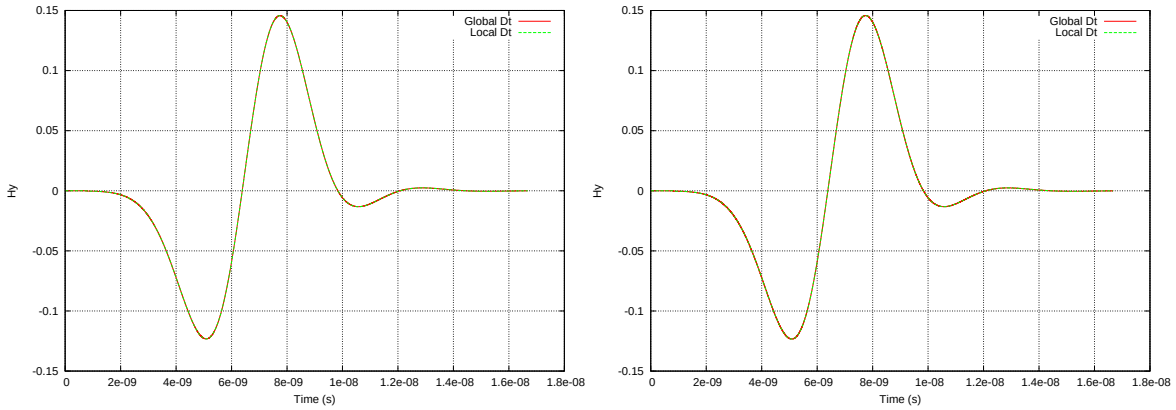


Figure 3.31 – Wave initiated by a localized source. Unstructured triangular mesh.



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.32 – Wave initiated by a localized source. Time evolution of E_z for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods (simulation time T=1.66×10^{-8} s).

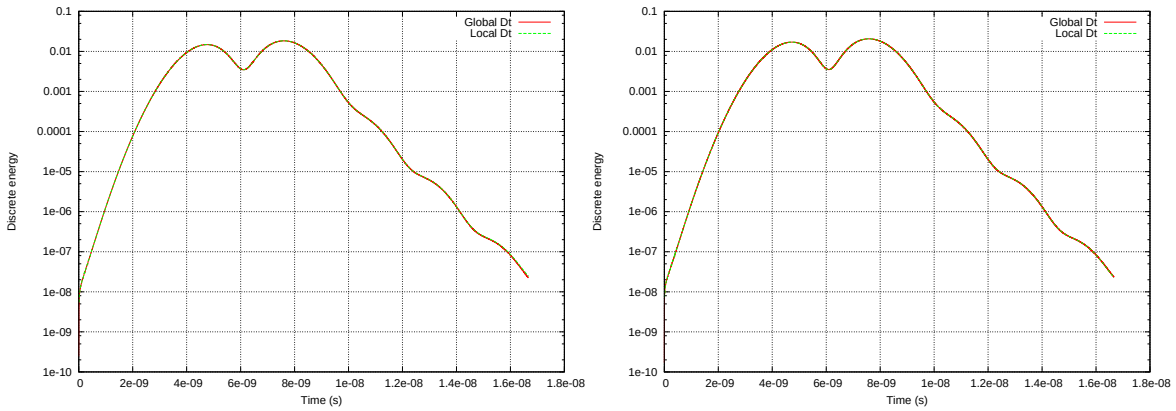


DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.33 – Wave initiated by a localized source. Time evolution of H_y for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods (simulation time T=1.66×10^{-8} s).

3.5.3 Scattering by an airfoil profile

We finally consider a test problem which allows for an assessment of the computational performances of the local time step based DGTD-$\mathbb{P}_p$ method. This problem consists in the scattering by an airfoil profile of a wave initiated by a localized source. For this test problem we make use of an unstructured triangular mesh consisting of 1505 vertices and 2842 triangles (see Fig. 3.35) which is refined ahead of the leading edge where the source is localized. Contour lines of the E_z components at various physical times are shown on Fig. 3.36. The ratio between the maximum and minimum values of the local time step is approximately equal to 662. For the local time stepping strategy the partitioning of the mesh is such that the fine and coarse parts respectively consist of 120 and 2722 elements. The corresponding ratio between the minimum time steps of the coarse (Δt_c) and fine (Δt_f) submeshes is approximately equal to 114.6 consequently we have fixed $p = 116$ in the local time stepping strategy and the time step Δt_f is redefined accordingly as $\Delta t_f = \Delta t_c / p$. The simulation time has been fixed to T=1.66×10^{-8} s. The time evolution of the E_z component at a fixed point of the domain, for the global and local time stepping strategies, is shown on Fig. 3.37, while the time evolution of the discrete energy is shown on Fig. 3.38. We note that, contrary to what has been concluded for the previous 2D problems, the numerical solutions resulting



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.34 – Wave initiated by a localized source. Time evolution of the discrete energy for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods (simulation time T=1.66×10^{-8} s).

from the global and local time stepping strategies perfectly match. On Fig. 3.37 we observe an amplitude error while there is no phase error. This behaviour deserves some further investigations, in particular to assess a possible dependence on the configuration of the local refinement in the underlying mesh.

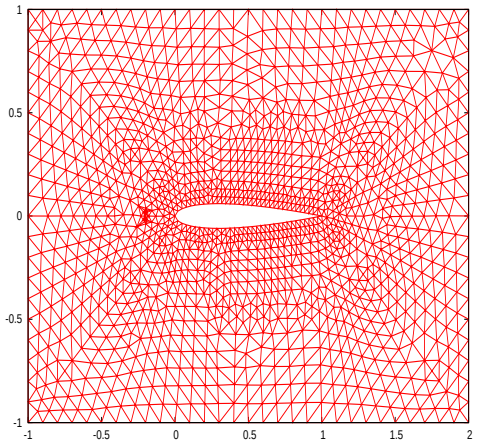
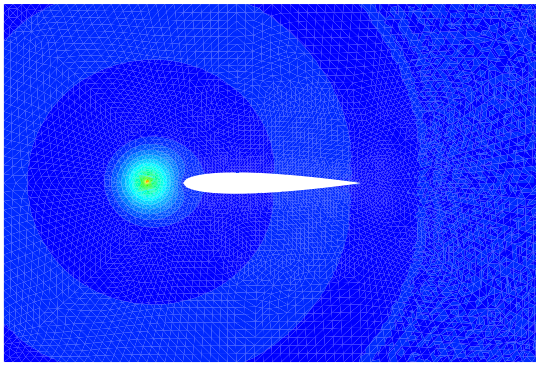


Figure 3.35 – Scattering by an airfoil profile. Unstructured triangular mesh.

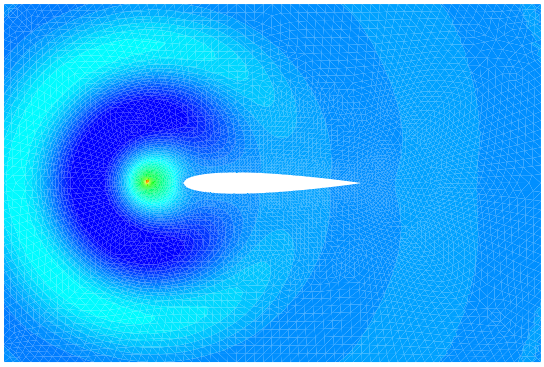
3.6 Conclusion

We have described a local time stepping strategy combined to a DGTD method for the solution of the system of time-domain Maxwell equations. The method presented here is based on a second order leap-frog scheme, therefore, a first natural sequel of this work is the extension of the proposed strategy to higher order time schemes and in particular to a fourth order leap-frog scheme. Preliminary results have been presented for 1D problems. The treatment of 2D problems is underway as well as a theoretical assessment of the stability of the method.

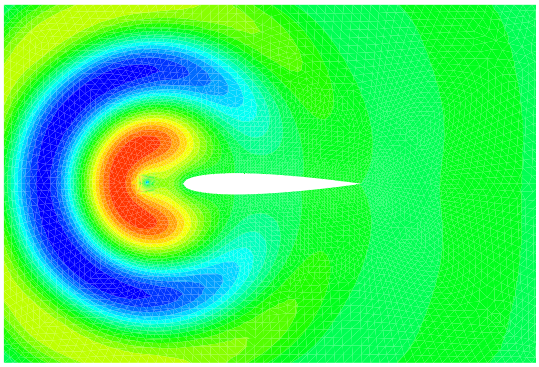
Acknowledgment. This work has been supported in part by CEA DAM (Military Application Division), CESTA center, Le Barp, under grant No. 4600187279.



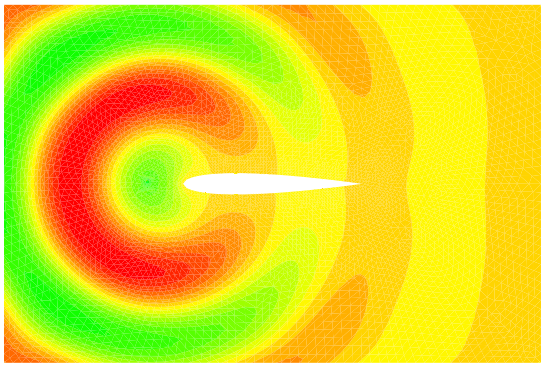
Time $= 5 \times 10^{-9}$ sec



Time $= 7.5 \times 10^{-9}$ sec



Time $= 8.33 \times 10^{-9}$ sec

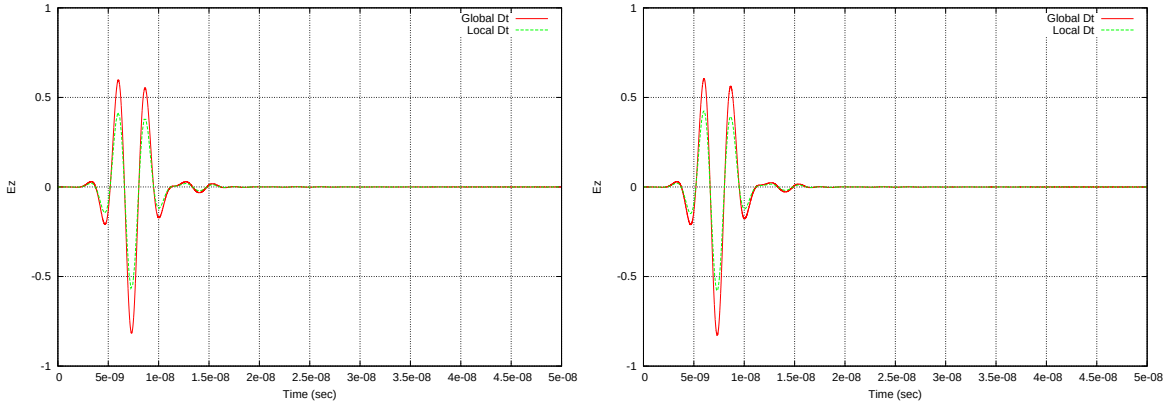


Time $= 9.16 \times 10^{-9}$ sec

Figure 3.36 – Scattering by an airfoil profile. Contour lines of the E_z component at various physical times. Simulations with the local time step based DGTD-$\mathbb{P}_2$ method.

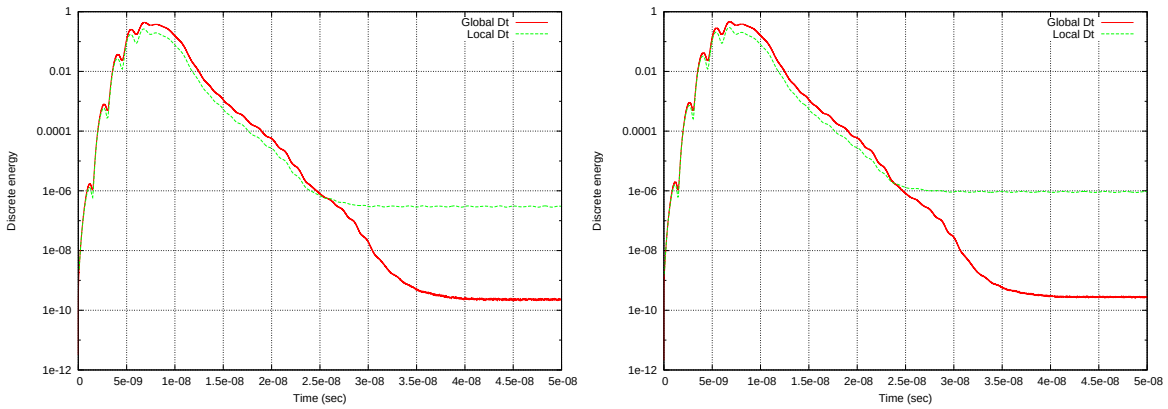
Table 3.5.5 – Scattering by an airfoil profile. Performance results.

Time stepping strategy	# iter DGTD-$\mathbb{P}_1$	CPU (s) DGTD-$\mathbb{P}_1$	# iter DGTD-$\mathbb{P}_2$	CPU (s) DGTD-$\mathbb{P}_2$
Global Δt	615,992	1220.0	5376	125.0
Local Δt	307,996	282.0	2688	27.0



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.37 – Scattering by an airfoil profile. Time evolution of E_z for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods (simulation time T=5×10^{-7} s).



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

Figure 3.38 – Scattering by an airfoil profile. Time evolution of the discrete energy for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods (simulation time T=5×10^{-7} s).

Chapitre 4

Multi-GPU acceleration of a DGTD method

Sommaire

4.1 Introduction

Discontinuous Galerkin (DG) methods have been the subject of numerous research activities in the last 15 years and have been successfully developed for various physical contexts modeled by elliptic, mixed hyperbolic-parabolic and hyperbolic systems of partial differential equations (PDEs). Indeed, DG methods offer a great flexibility in the mesh design by allowing arbitrary unstructured geometries and irregular non-conforming meshes [Fahs 2009]. Another advantage of DG methods is the freedom of choosing the elemental polynomial degrees without the need to enforce any conformity requirement which, combined with the possibility of anisotropic mesh refinement, makes them particularly suited in the context of *hp*-adaptivity. Additionally, Discontinuous Galerkin Time Domain (DGTD) methods lead to (block-) diagonal mass matrices and therefore yield fully explicit, inherently parallel methods when coupled with explicit time stepping. Moreover, continuity is weakly enforced across mesh interfaces by adding suitable bilinear forms (often referred as numerical fluxes) to the standard variational formulations. As

they are able to obtain very accurate spatial discretizations on arbitrary unstructured meshes with inexpensive explicit time stepping algorithms and allow an easy control of accuracy without compromising simulation stability, DGTD methods are now widely used for the solution of wave propagation problems in which low dispersion errors are a necessary condition for accuracy. DGTD methods for solving the time-domain Maxwell equations have been proposed by several authors [Hesthaven 2002] - [Fezoui 2005] - [Cohen 2006]. Despite the achievements so far, it seems that a major limitation to a wider adoption of DGTD methods for large-scale applications in various physical domains and especially for electromagnetic wave propagation problems, is their excessive overhead in terms of computational time and memory occupancy. Not surprisingly, several ongoing research efforts on discontinuous Galerkin methods aim at improving their efficiency both from the numerical and computational point of view. The present work is a contribution to this problematic and we concentrate here on the exploitation of massively parallel computing systems based on graphical processing unit (GPU) acceleration cards to make the most of the inherent locality and parallelism of DGTD methods.

For more than two decades, the computer industry has witnessed an ever increasing drive of performance improvement. Speeding up processor frequency has considerably improved the processing capabilities given in Giga Floating point operations per second (GFlops), until physical limits of semiconductor based microelectronics have become a major design concern. In the recent years, design issues for more capable microprocessors have evolved towards the development of multi-core CPUs effectively multiplying the potential performance with several separate processing cores on the same chip, used concurrently. One other particularly noteworthy approach is the use of many-core devices such as Graphical Processing Units (GPUs), accelerating computations orchestrated by a conventional CPU program. Efforts to exploit GPUs, for non-graphical applications have been underway since 2003 and has evolved into programmable and massively parallel computational units with very high memory bandwidth. From this time to the present day a review of research works aiming at harnessing GPUs for the acceleration of partial differential equations solvers would hardly fit into one page. In particular, the development of GPU enabled high order numerical methods for the solution of partial differential equations is a rapidly growing field. Focusing on contributions that are dealing with wave propagation problems, GPUs have been considered for the first time for computational electromagnetics and computational geoseismics applications respectively by Klöckner et al. [Klöckner 2009] and by Komatitsch et al. [Komatitsch 2009] - [Komatitsch 2010b] - [Komatitsch 2010a]. The present work shares several concerns with [Klöckner 2009] which describes the development of a GPU enabled discontinuous Galerkin (DG) method formulated on an unstructured tetrahedral mesh for the discretization of hyperbolic systems of conservation laws. Computational results are presented for the solution of the system of 3D time-domain Maxwell equations. As it is the case with the DG method considered in [Klöckner 2009], the approximation of the unknown field in a tetrahedron relies on a high order nodal interpolation method which is a key feature in view of exploiting the processing capabilities of the GPU architecture. The main design differences are found in the adopted numerical flux (Klöckner et al. consider an upwind flux while we make use of a centered flux) and time-stepping scheme (a 4 steps Runge-Kutta scheme is considered in [Klöckner 2009] while we rely on a second order leap-frog scheme). In [Klöckner 2009], the CUDA adaptation of the numerical kernels of the studied DG method is detailed, emphasizing the data and computation layout leading to highly tuned implementations of these kernels. Single precision performance on a NVIDIA GTX 280 GPU exceeds 250 GFlops for a 9th order interpolation while the speedup relatively to a CPU calculation is maximized for a 4th order interpolation (the observed speedup is 65 in this case).

A recent evolution of this work described in [Klöckner 2009] is presented in Gödel et al. [Gödel 2010] where the authors discuss the adaptation of a multirate time stepping based DG method for solving the time-domain Maxwell equations on a multiple GPU system. Load balancing issues linked to the local time stepping scheme are treated by a weighted graph partitioning. Performance results are presented for simulations conducted on a system with 4 GPUs but the weak and strong scalability of the resulting hybrid CPU-GPU DG method are not studied. Another widely adopted high order method, namely the spectral element method, is considered in [Komatitsch 2009] - [Komatitsch 2010b] - [Komatitsch 2010a]

for the simulation of seismic wave propagation. In [Komatitsch 2010a], the authors discuss the implementation of the SPECFEM3D software on a large cluster of NVIDIA GT200 GPUs using the CUDA environment and non-blocking message-passing using MPI. Speedups as high as 25 are obtained between the CUDA+MPI version and the highly optimized C+MPI original version of the SPECFEM3D software. A similar hybrid SIMD-MIMD parallelization strategy is considered in the present study.

We present in this chapter a high performance computing methodology for the simulation of electromagnetic wave propagation in complex domains and heterogeneous media. Our objective is to demonstrate the potential of the proposed numerical methodology by exploiting the inherent DGTD parallelism combined with the GPU computing capabilities for the study of realistic wave propagation in biological tissues and its application to the numerical evaluation of radio frequency absorption in head tissues, as they are exposed to radiation from a cellular phone, and for the simulation of scattering of a plane wave by an aircraft geometry. The combination of these brand features will result in significant increase in efficiency and reduction of the computational time. For this purpose, we consider a DGTD method on tetrahedral meshes for solving the 3D time-domain Maxwell equations [Fezoui 2005] - [Bernacki 2006b] - [Bernacki 2006a] and extend it to modern parallel computing systems with multiple GPU acceleration cards by adopting a hybrid strategy that combines a coarse grain SPMD programming model for inter-GPU parallelization and a fine grain SIMD programming model for intra-GPU parallelization. The performance improvement will be demonstrated through large-scale simulations that are performed on a cluster of GPUs using realistic heterogeneous models of head tissues built from medical images and an aircraft geometry involving more than two millions elements. Particular attention will have to be paid to data distribution (geometrical entities of the mesh : vertices, elements, and faces) at stake in the main calculation loops. Performance tuning will consist in balancing conflicting goals, taking into account kernels sizes, memory access patterns, sizes of small on-chip memories, multiprocessor occupancy, etc. Crucial aspects such as data movement between the different types of memory (RAM, registers, caches, etc.), parallelism, and load balancing will have to be taken in consideration in a unified way to efficiently solve the data mapping problem.

The sequel of the chapter is organized as follows : the algorithmic adaptation for obtaining a GPU-enabled implementation of the DGTD method based on CUDA is discussed in section 4.2 ; section 4.3 is devoted to the application of the resulting numerical methodology to the calculation of the electromagnetic wave propagation in realistic geometrical models of the head tissues, as well as to the analysis of the weak and strong scalability properties on a cluster of GPUs ; section 4.4 extends the multi-GPU parallelization capabilities by the adaptation and tuning of the DGTD method on the latest generation of CUDA architecture ; finally, section 4.5 concludes this work.

This chapter is an enhanced version of the research report [Cabel 2011].

4.2 Implementation on GPU clusters

We discuss in this section the design principles that we have adopted for the implementation of the DGTD-$\mathbb{P}_{p_i}$ method described in section 1.2 of chapter . Prior to do so, we refer to section A.1 of annex A for a detailed description of the main aspects of GPU computing as well as basic features of the CUDA programming model.

4.2.1 CUDA-enabled DGTD kernels

We recall here the local weak formulation of the DGTD-$\mathbb{P}_{p_i}$ method involving volume integrals over an element τ_i and surface integrals over an element boundary $\partial\tau_i$:

$$\left\{\begin{array}{rcl} \displaystyle\int_{\tau_i} \vec{\varphi}\cdot\epsilon_i\partial_t\vec{\mathbf{E}}_i & = & \displaystyle\frac{1}{2}\int_{\tau_i}(\mathrm{curl}\vec{\varphi}\cdot\vec{\mathbf{H}}_i+\mathrm{curl}\vec{\mathbf{H}}_i\cdot\vec{\varphi})-\frac{1}{2}\sum_{k\in\mathcal{V}_i}\int_{a_{ik}}\vec{\varphi}\cdot(\vec{\mathbf{H}}_k\times\vec{n}_{ik}), \\ \displaystyle\int_{\tau_i} \vec{\varphi}\cdot\mu_i\partial_t\vec{\mathbf{H}}_i & = & \displaystyle-\frac{1}{2}\int_{\tau_i}(\mathrm{curl}\vec{\varphi}\cdot\vec{\mathbf{E}}_i+\mathrm{curl}\vec{\mathbf{E}}_i\cdot\vec{\varphi})+\frac{1}{2}\sum_{k\in\mathcal{V}_i}\int_{a_{ik}}\vec{\varphi}\cdot(\vec{\mathbf{E}}_k\times\vec{n}_{ik}), \end{array}\right. \tag{4.2.1}$$

which leads to the formulation of the local system of ordinary differential equations for each τ_i :

$$\left\{\begin{array}{rcl} \displaystyle M_i^{\epsilon}\frac{d\mathbf{E}_i}{dt} & = & \displaystyle K_i\mathbf{H}_i-\sum_{k\in\mathcal{V}_i}S_{ik}\mathbf{H}_k, \\ \displaystyle M_i^{\mu}\frac{d\mathbf{H}_i}{dt} & = & \displaystyle -K_i\mathbf{E}_i+\sum_{k\in\mathcal{V}_i}S_{ik}\mathbf{E}_k, \end{array}\right. \tag{4.2.2}$$

where the symmetric positive definite mass matrices M_i^{η} (η stands for ϵ or μ), the symmetric stiffness matrix K_i (both of size $d_i\times d_i$) and the symmetric interface matrix S_{ik} (of size $d_i\times d_j$ in the general case) are given by :

$$\left\{\begin{array}{rcl} (M_i^{\eta})_{jl} & = & \displaystyle\eta_i\int_{\tau_i}{}^t\vec{\varphi}_{ij}\cdot\vec{\varphi}_{il}, \\ (K_i)_{jl} & = & \displaystyle\frac{1}{2}\int_{\tau_i}{}^t\vec{\varphi}_{ij}\cdot\mathrm{curl}\vec{\varphi}_{il}+{}^t\vec{\varphi}_{il}\cdot\mathrm{curl}\vec{\varphi}_{ij}, \\ (S_{ik})_{jl} & = & \displaystyle\frac{1}{2}\int_{a_{ik}}{}^t\vec{\varphi}_{ij}\cdot(\vec{\varphi}_{kl}\times\vec{n}_{ik}). \end{array}\right.$$

We now describe the implementation strategy adopted for GT200 and Fermi generations of NVIDIA GPUs and for calculations in single precision floating point arithmetic. We first note that the main computational kernels of the DGTD-$\mathbb{P}_{p_i}$ method considered in this study are the volume and surface integrals over τ_i and $\partial\tau_i$ appearing in (4.2.1). Moreover, we limit ourselves to a uniform order method *i.e.* $p\equiv p_i$ is the same for all the elements of the mesh, and we present experimental results for the values $p=1,2,3,4$. At the discrete level, these local computations translate into the matrix-vector products appearing in (4.2.2). The discrete equations for updating the electric and magnetic fields are composed of the same steps and only differ by the fields they are applied to. They both involve the same kernels that we will refer to in the sequel as `intVolume` (computation of volume integrals), `intSurface` (computation of surface integrals) and `updateField` (update of field components). All these kernels stick to the following paradigm :

1. Load data from device memory to shared memory
2. Synchronize with all the other threads of the block so that each thread can safely read shared memory locations that were populated by different threads
3. Process the data in shared memory
4. Synchronize again to make sure that shared memory has been updated with the results
5. Write the results back to device memory

This paradigm ensures that almost all the operations on data allocated in global memory are performed in a coalesced way.

In our implementation, some useful elementary matrices, such as the mass matrix computed on the reference element, are stored in constant memory because they are small and are accessed following constant memory patterns.

For the sequel, we introduce the following notations :

- NBTET is the number of tetrahedra that are treated by a block of threads. It depends of the chosen interpolation order and it is taken to be a multiple of 16 (32 for Fermi) because of the way one loads and writes data to and from device memory.
- NDL is the number of degrees of freedom (d.o.f.) in an element τ_i for each field component, for a given interpolation order.
- NDF is the number of d.o.f. on a face a_{ik} for each field component, for a given interpolation order.

Initial implementations for CPU based systems have been specifically adapted for GPU computing. The structure of the CUDA adaptation of the main serial time loop of the DGTD method is given on figure A.20 in section A.2 of annex A and we refer to section A.3 of annex A for the algorithmic description of GPU implementation on both NVIDIA GT200 and Fermi families. We give below a short description of the way the different kernels have been built.

4.2.1.1 Volume integral kernel : `intVolume`

The Volume integral kernel `intVolume` loops over mesh elements τ_i for computing the volume integral :

$$\mathbf{F}_{\tau_i} = \frac{1}{2} \iiint_{\tau_i} (\nabla \times \vec{\varphi} \cdot \vec{\mathbf{H}}_i + \nabla \times \vec{\mathbf{H}}_i \cdot \vec{\varphi}) \, \mathrm{d}\omega, \tag{4.2.3}$$

and updates flux balance of elements τ_i.

This kernel operates either on each d.o.f. of a tetrahedron on GT200, either on two d.o.f. of one tetrahedron on Fermi. Since the number of d.o.f. increases with the interpolation degree, resources needed by this kernel (registers and shared memory) also raise. Consequently, we wrote two versions of this kernel : one kernel for $p = 1$ and 2, and the other one for $p = 3$ and 4. However, these two versions have some common features. First, each thread computes the same number of d.o.f. of one tetrahedron (either one or two, depending on architectures). The second common feature is the data stored in shared memory, which are the field and the flux balance components and some geometrical quantities associated to a tetrahedron. The last common feature is the number of tetrahedra operated by a block (*i.e.* NBTET). The main difference is that when in the low order version a block computes all the d.o.f. of the NBTET tetrahedra (*i.e.* NDL on GT200 and NDL/2 on Fermi), the high order volume kernel only computes a certain number of d.o.f. of the NBTET tetrahedra. Consequently, in the latter case, two or three instances of the kernel are necessary to compute all the d.o.f. of all the tetrahedra. This approach induces a drawback because we have to load field data in two or three kernels instead of one. Indeed, the dimension of a block is NBTET*NDL on GT200 (NBTET*NDL/2 on Fermi) which leads to blocks of more than 512 threads on GT200 (1024 threads on Fermi) for high interpolation orders which is not possible in CUDA. However, there is also a benefit because computing a lower number of d.o.f. in a kernel allows us to use less shared memory in the buffer storing field data and less registers in a kernel thus increasing the occupancy of the GPU.

4.2.1.2 Surface Integral kernel : `intSurface`

The Surface integral kernel `intSurface` loops over mesh elements at stake in all faces $a_{ik} = \tau_i \cap \tau_j$ for computing the surface integral :

$$\mathbf{F}_{a_{ik}} = -\mathbf{F}_{a_{ki}} = \frac{1}{2} \sum_{k \in \mathcal{V}_i} \iint_{a_{ik}} \vec{\varphi} \cdot (\vec{\mathbf{H}}_k \times \vec{n}_{ik}) \, \mathrm{d}s,$$

and update flux balance of element τ_i and τ_j, which implies redundant calculation of $\mathbf{F}_{a_{ik}}$ and $\mathbf{F}_{a_{ki}}$ (initially we looped over all faces to circumvent this drawback). Fictive neighbor fields are therefore created to process boundary faces as internal faces.

For this kernel, one thread works either on one surface d.o.f. of one tetrahedron on GT200, either on two surface d.o.f. of one tetrahedron on Fermi (for $p \geq 2$). Similarly to the `intVolume` kernel, two versions

of this kernel have been implemented. For the low order version, a thread applies the influence of its d.o.f. to the four faces of its tetrahedron whereas for the high order version, a thread only works on one face of its tetrahedron. So, for the low order version, a block computes the numerical flux for four faces of NBTET tetrahedra instead of one face of NBTET tetrahedra for the high order version. Therefore, the high order version has to launch four kernels instead of one for the low order version. Here, we work on the surface d.o.f. (NDF) but fields components are store using the volume d.o.f. (NDL) so we need to use a permutation matrix to link these different local numberings of these d.o.f. Moreover, a face of a tetrahedron is also shared by another tetrahedron and the corresponding field values are needed in the computation of the elementary flux. Consequently, we cannot load field data in a coalesced way and we have to use texture memory. Field values are loaded before each face computation. Nevertheless, the high order version has a memory drawback compared to the lower one. Indeed, because there are four launches of the function, data are written four times to the flux table instead of once in the low order version.

4.2.1.3 Update kernel : `updateField`

The Update kernel `updateField` loop over mesh elements τ_i and updates the electromagnetic field by exploiting flux balance of element τ_i.

There are four update kernels. First of all, update kernels are a bit different according to the field they are working on (electric or magnetic). Since in this case a thread works either on one d.o.f. of a tetrahedron on GT200, either on two d.o.f. of one tetrahedron on Fermi, the dimension of a block is NBTET*NDL on GT200 and NBTET*NDL/2 on Fermi. Consequently, as for the `intVolume` kernel, we need a special version for the higher interpolation orders in order to avoid exceeding the maximum number of threads per block. In the high order version, we adopt an approach where a thread deals with two different d.o.f. of a tetrahedron which allows a block to compute all the d.o.f. for NBTET tetrahedra. This approach is less efficient for the lower interpolation orders. The two versions of the electric field update kernels need only one shared memory table. Indeed, in the first step, the flux computed by the previous kernels is loaded in this table, used to do some computations and then stored in a register. Therefore, the shared memory table is no longer used at the end of this part. In the second step, we load the previous values of the electric field in it in a coalesced way. In a third step, we update the value of the field in the shared memory, and in the last step, we write the new value of the field in the global memory. The update of the magnetic field follows the same pattern as the update of the electric field.

4.2.2 Multi-GPU parallelization strategy

The multi-GPU parallelization strategy adopted in this study combines a coarse grain SPMD model based on a partitioning of the underlying tetrahedral mesh, with a fine grain SIMD model through the development of CUDA enabled DGTD kernels. A non-overlapping partitioning of the mesh is obtained using a graph partitioning tool such as MeTiS or Scotch and results in the definition of a set of sub-meshes. The interface between neighboring sub-meshes is a triangular surface. In the current implementation of this strategy, there is a one to one mapping between a sub-mesh and a GPU. Then the CUDA kernels described previously are applied at the sub-mesh level. The operations of the DGTD method are purely local except for the computation of the numerical flux for the approximation of the boundary integral over $\partial\tau_i$ in (4.2.1) which requires, for a given element, the values of the electromagnetic field components in the face-wise neighboring elements. For those faces which are located on an interface between neighboring sub-meshes, the availability of the electromagnetic field components on the attached elements is obtained thanks to point-to-point communications implemented using non-blocking MPI send and receive operations in order to overlap as much as possible communication operations by local computations of the volume integrals in (4.2.1). Moreover, we also overlap most of the PCI-Express communications by using a CudaHostAlloc buffer mapped into the CUDA address space which allows us

to let the driver manage the CPU-GPU communication (see A.1.3) of the electromagnetic field between MPI subgrids.

4.3 Performance assessments on GT200 GPUs

4.3.1 Hardware configuration

GPU timings (for all the performance results presented here and in the following subsections) are for single precision arithmetic computations and include data transfer operations from the CPU memory to the GPU device memory prior to the time stepping loop, and vice versa at the end of the time stepping loop. Numerical simulations have been performed on a hybrid CPU-GPU cluster localized at the CEA-CCRT (Research and Technology Computing Center) with 1068 Intel CPU nodes and 48 Tesla S1070 1U GPU computing systems. Each Tesla S1070 has four GT200 GPUs (C1060) with 4 Go memory each and two PCI-Express 2.0 buses. The NVIDIA Tesla S1070 computing system with 960 cores in a 1U chassis, delivers 4 TFlops of peak single precision floating point performance (310 GFlops in double precision), 409.6 GB/s (4×102.4 GB/s) of maximum bandwidth and includes 16 GB of ultra-fast memory. The Tesla systems are connected to BULL Novascale R422 E1 nodes each composed of two quad-core Intel Xeon X5570 Nehalem processors operating at 2.93 GHz, themselves connected by an InfiniBand DDR (Double Data Rate) network, with a 20 Gb/s speed and 24 Go of memory.

4.3.2 Performance results on a model test problem

We first present results for the assessment of the weak scalability properties of the GPU-enabled DGTD-$\mathbb{P}_p$ method. For that purpose, we consider a model test problem which consists in the propagation of a standing wave in a perfectly conducting unitary cubic cavity. For this simple geometry, we make use of regular uniform tetrahedral meshes respectively containing 3,072,000 elements for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods, 1,296,000 elements for the DGTD-$\mathbb{P}_3$ method and 750,000 elements for the DGTD-$\mathbb{P}_4$ method for the experiments involving one GPU. As usual in the context of a weak scalability analysis, the size of each mesh is increased proportionally to the number of computational entities. Moreover, since these meshes are regular discretizations of the cube, it is possible to construct perfectly balanced partitions and this is achieved here by constructing the tetrahedral meshes in parallel (*i.e.* on a subdomain basis) given a box-wise decomposition of the domain. The graphs of Fig. 4.1 illustrate the almost perfect weak scalability of the GPU enabled DGTD-$\mathbb{P}_p$ method with $p = 1, \ldots, 4$ for up to 128 GPUs. GFlops rates are plotted on the graphs of Fig. 4.2 while Table 4.3.1 summarizes the measured timing measures for 1000 iterations of the leap-frog time scheme (1.2.16), and corresponding GFlops rates for 1 and 128 GPUs. These results illustrate an almost perfect weak scalability of the GPU enabled DGTD-$\mathbb{P}_p$ method with $p = 1, \ldots, 4$ for up to 128 GPUs. It also appears from these results that, for the proposed GPU implementation of the DGTD-$\mathbb{P}_p$ method and the hardware configuration considered in the above numerical experiments, the third-order scheme yields the best performance while, when increasing further the interpolation order, the sustained performance decreases due to bandwidth-bound effects.

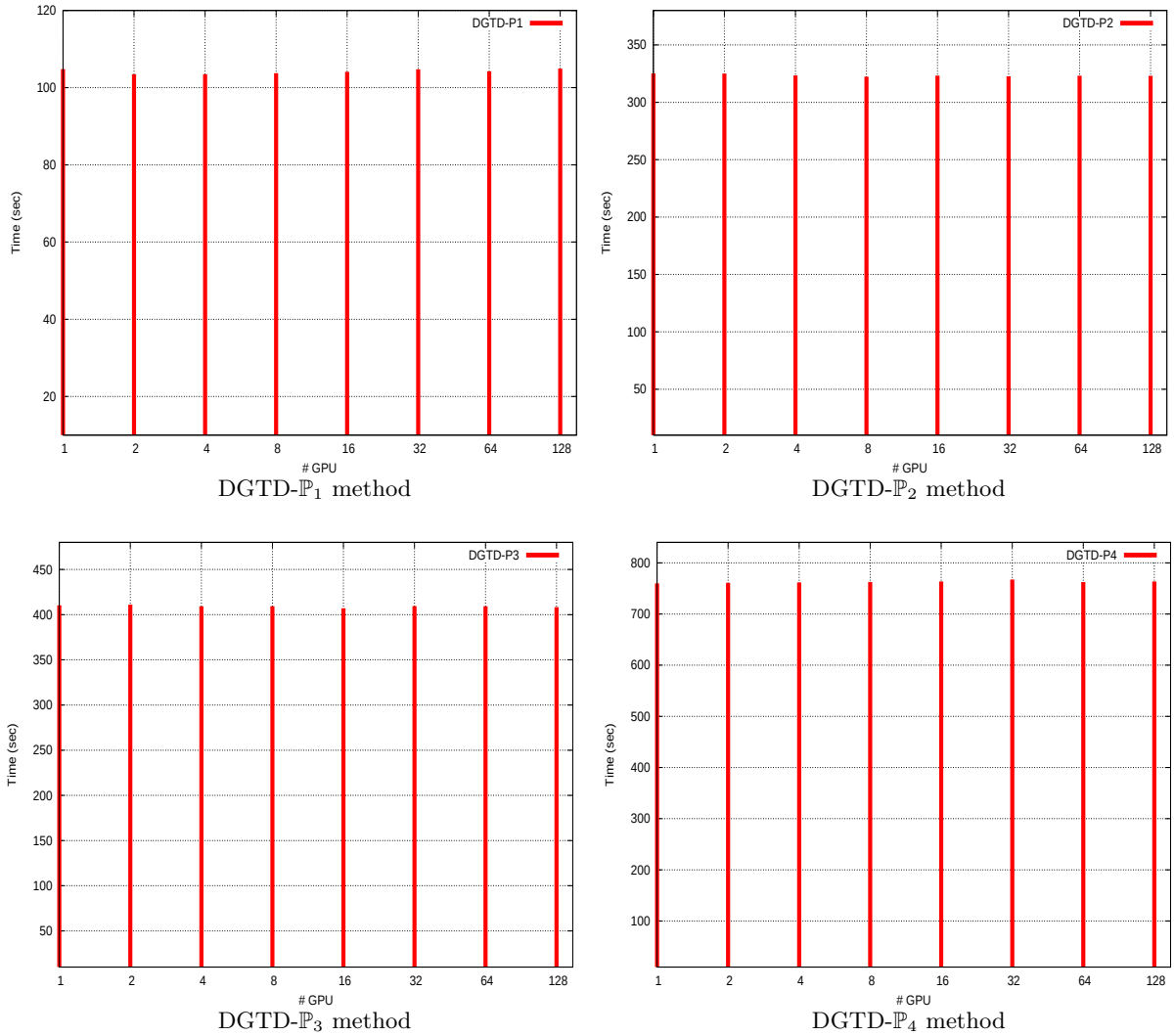
4.3.3 Performance results on a realistic problem

The objective of this section is to assess the strong scalability of the GPU-enabled DGTD method by considering a realistic wave propagation problem involving a heterogeneous medium with irregularly shaped boundaries. More precisely, we demonstrate the application of the GPU-enabled DGTD based numerical methodology to the calculation of the electromagnetic wave propagation in realistic geometrical models of the head tissues As a matter of fact, numerical modeling is nowadays increasingly used and progressively becoming a mandatory path for the study of the interaction of electromagnetic fields with biological tissues. This is in particular the case for the evaluation of the SAR (Specific Absorption Rate)

# GPU	DGTD-$\mathbb{P}_1$		DGTD-$\mathbb{P}_2$	
	Time	GFlops	Time	GFlops
1	104.7 sec	63	325.1 sec	92
128	104.9 sec	8072	323.1 sec	11844

# GPU	DGTD-$\mathbb{P}_3$		DGTD-$\mathbb{P}_4$	
	Time	GFlops	Time	GFlops
1	410.3 sec	106	759.8 sec	94
128	408.4 sec	13676	763.6 sec	12009

TABLE 4.3.1 – Weak scalability assessment : timings and sustained performance figures.



DGTD-$\mathbb{P}_1$ method

DGTD-$\mathbb{P}_2$ method

DGTD-$\mathbb{P}_3$ method

DGTD-$\mathbb{P}_4$ method

FIGURE 4.1 – Weak scalability assessment : evolution of the computing time with the number of GPUs.

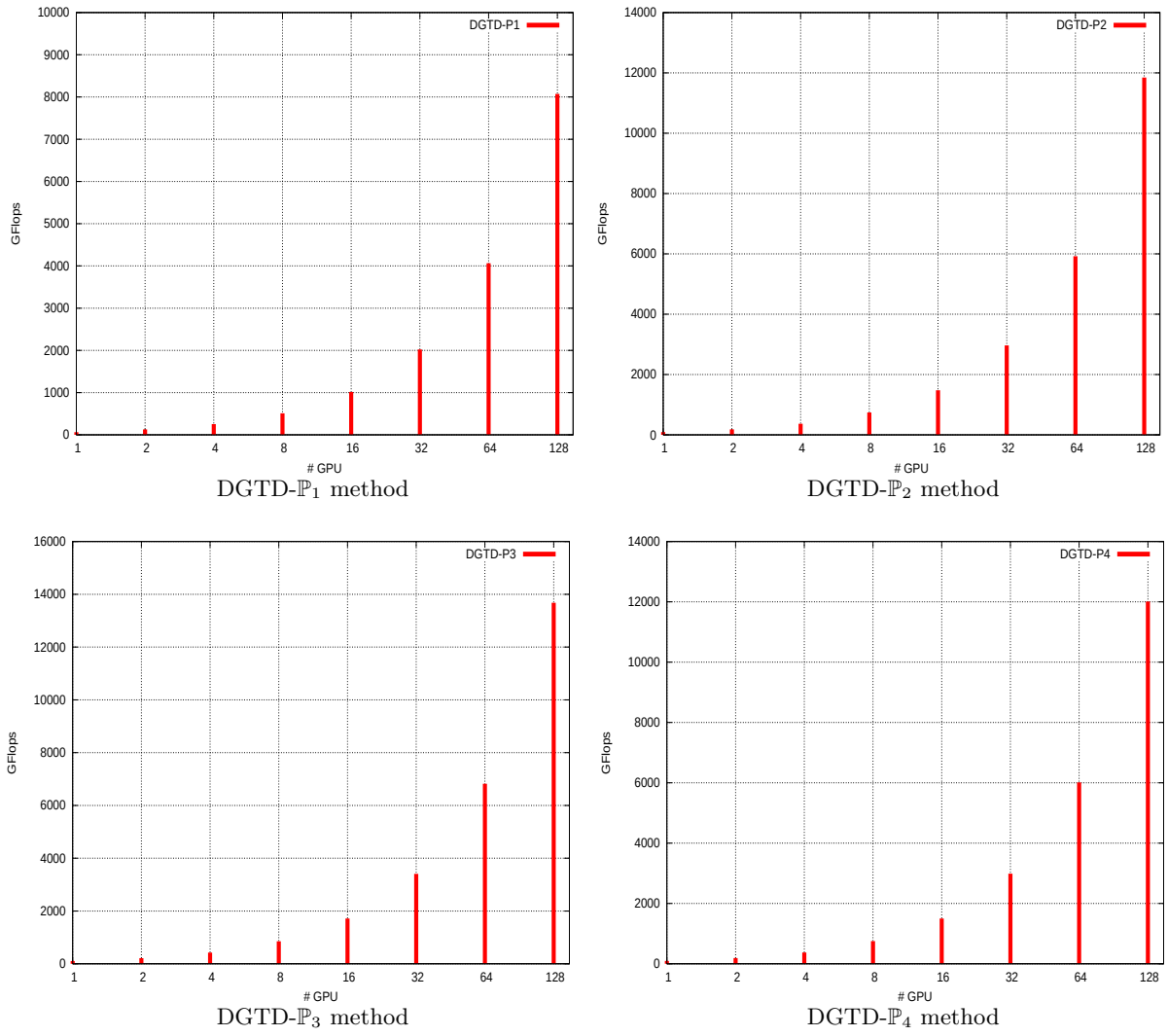


Figure 4.2 – Weak scalability assessment : evolution of GFlops rates with the number of GPUs.

which is a measure of the rate at which electric energy is absorbed by the tissues when exposed to a radio-frequency electromagnetic field. The SAR is defined as the power absorbed per mass of tissue and has units of watts per kilogram. Numerical SAR calculations are intensively considered for dosimetry studies of the exposure of human tissues to microwave radiations from wireless communication systems [Bernardi 2001] - [Gandhi 2001] - [Conil 2008] to cite but a few of many examples. These studies are useful for assessing the possible thermal effects (temperature rise in tissues resulting from electric energy dissipation) as well as for compliance testing to regulatory limits. SAR calculations are also relevant for certain medical applications such as for the design of microwave hyperthermia systems [Camart 1996] - [Dunn 1996] - [Lin 2000], and for the design of micro-antennas to be implanted inside the human body [Kim 2004].

Despite the high complexity both in terms of heterogeneity and geometrical features of tissues, the great majority of numerical dosimetry studies have been conducted using the widely known Finite Difference Time Domain (FDTD) method due to Yee [Yee 1966]. In this method, the whole computational domain is discretized using a structured (Cartesian) grid. Due to the possible straightforward implementation of the algorithm and the availability of computational power, FDTD is currently the leading method for numerical assessment of human exposure to electromagnetic waves. In the particular case of mobile phone radiation, the FDTD method is applied to heterogeneous discretized models of human head tissues built from medical images. Thus, the grid generation process is highly simplified since the voxel based image can be used at a minimal effort as the computational grid for the FDTD method. However it is well known that albeit being highly flexible and second-order accurate in a homogeneous medium, the Yee scheme suffers from serious accuracy degradation when used to model curved objects or when treating material interfaces.

In an attempt to offer an alternative numerical dosimetry methodology which allows for a realistic modeling of geometrical features and tissue interfaces, we consider here the use of a discontinuous Galerkin method formulated on non-uniform tetrahedral meshes.

4.3.3.1 Numerical treatment of biological propagation media

Human tissues are dispersive materials and thus require a specific treatment when modeling time-domain problems. However, in this study, we do not take into account the dispersive characteristic of the propagation media and simply consider them as conductive as it is done in the great majority of numerical dosimetry studies. Then, a conductive current $\vec{J}_\sigma = \sigma\vec{E}$ has to be taken into account in (3.2.1). It is straightforward to verify that the space discretization of this current term in the framework of the discontinuous Galerkin formulation described in subsection 1.2 leads to the introduction of the term $-M_i^\epsilon \mathbf{E}_i$ in the right hand side of the first equation of (4.2.2). Then this term is time integrated as $-M_i^\epsilon \left(\mathbf{E}_i^n + \mathbf{E}_i^{n+1}\right)/2$ meaning that both the left and right hands side terms of (1.2.16) are affected by the discretization of this conductive current.

4.3.3.2 Tetrahedral mesh based geometric models of head tissues

The DGTD-$\mathbb{P}_{p_i}$ method described previously assumes that the computational domain is discretized using tetrahedral elements. In this study, we aim at exploiting this numerical method for the calculation of the SAR induced in head tissues. A first step is thus to construct compatible geometrical models of the head tissues. Starting from magnetic resonance images of the Visible Human 2.0 project [Ratiu 2003], head tissues are segmented and surface triangulations of a selected number of tissues are obtained. Different strategies can be used in order to obtain a smooth and accurate segmentation of head tissues and interface triangulations as well. The strategy adopted in this work consists in using a variant of Chew's algorithm [Chew 1993], based on Delaunay triangulation restricted to the interface, which allows to control the size and aspect ratio of interface triangles [Boissonnat 2005]. Example of triangulations of the skin, skull and brain are shown on Fig. 4.3. In a second step, these triangulated surfaces together with a triangulation of the artificial boundary (absorbing boundary) of the overall computational domain are used as inputs

for the generation of volume meshes. The GHS3D tetrahedral mesh generator [George 1991] is used to mesh the volume domains between the various interfaces. The exterior of the head must also be meshed, up to a certain distance and the computational domain is artificially bounded by a sphere surface corresponding to the boundary Γ_a on which the Silver-Müller absorbing boundary condition (1.1.13) is imposed. Moreover, a simplified mobile phone model (metallic box with a quarter-wave length mounted on the top surface) is included and placed in vertical position close to the right ear (see Fig. 4.4). The surface of this metallic box defines the boundary Γ_m. Overall, the geometrical models considered here consist of four tissues (skin, skull, CSF - Cerebro Spinal Fluid and brain).

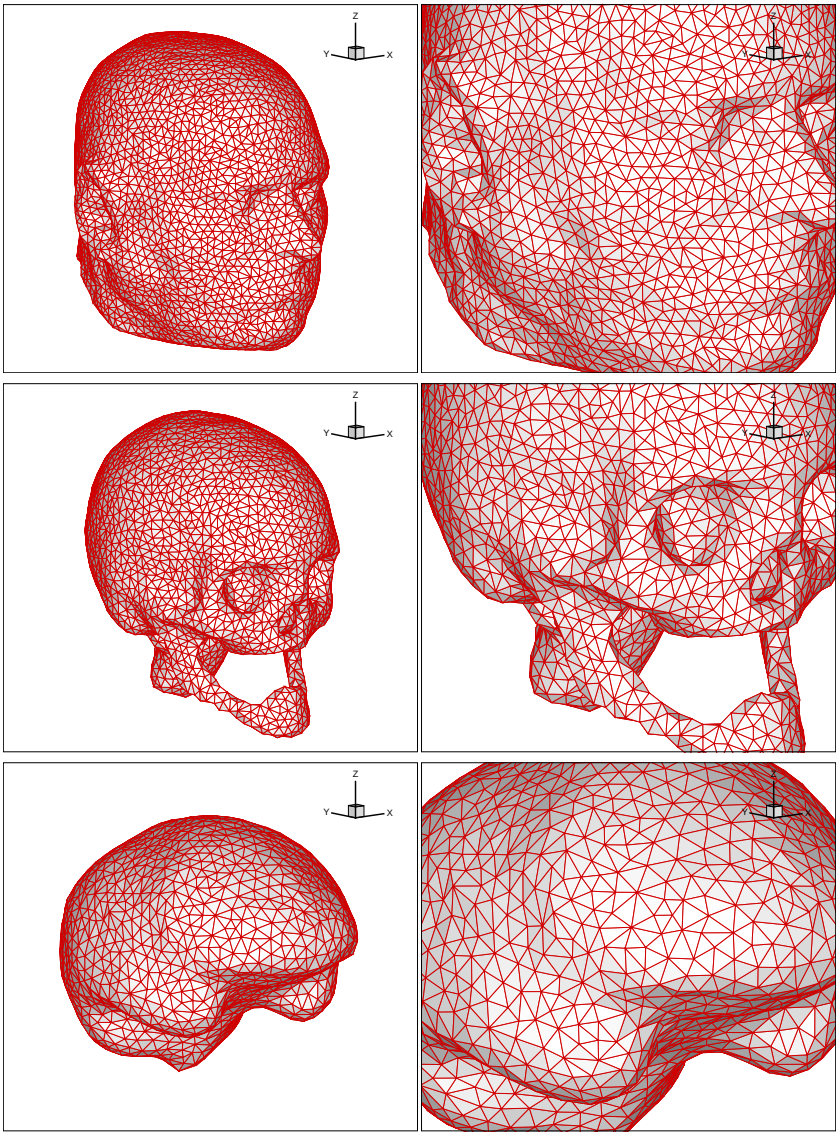


Figure 4.3 – Surface meshes of the skin, skull and brain.

4.3.3.3 Numerical results

All the numerical experiments reported here are concerned with the propagation of an electromagnetic wave emitted by a dipolar source localized (and centered) between the lower tip of the antenna and the top surface of the metallic box defining the mobile phone, and is modeled by a current of the form ($\vec{x}_d$

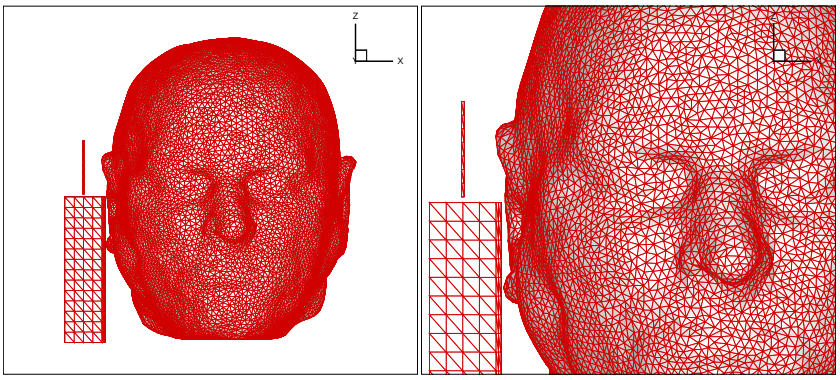


FIGURE 4.4 – Simplified mobile phone model and its positioning.

is the localization point of the source) :

$$\vec{J}_s(\vec{x},t) = \delta(\vec{x}-\vec{x}_d)f(t)\vec{e}_z, \tag{4.3.1}$$

where $f(t)$ is a sinusoidally varying temporal signal. This source current is easily introduced and discretized according to the discontinuous Galerkin formulation discussed in subsection 1.2. The physical simulation time has been fixed to 5 periods of the temporal signal of (4.3.1). A discrete Fourier transform of the components of the electric field is computed during the last period of the simulation. The characteristics of the tissues are summarized in Table 4.3.2 where the values of the electrical permittivity correspond to a frequency F=1800 MHz.

Tissue	ε_r	λ **(mm)**	σ **(S/m)**	ρ **(Kg/m^3)**
Skin	43.85	26.73	1.23	1100.0
Skull	15.56	42.25	0.43	1200.0
CSF	67.20	20.33	2.92	1000.0
Brain	43.55	25.26	1.15	1050.0

TABLE 4.3.2 – Exposure of head tissues to a localized source radiation : electromagnetic characteristics of tissues.

We consider a sequence of three unstructured tetrahedral meshes whose characteristics are summarized in Table 4.3.3. For these meshes, the artificial boundary Γ_a is a spherical surface approximately located one wavelength away from the skin. The tetrahedral meshes are globally non-uniform and the quantities L_{min}, L_{max} and L_{avg} in Table 4.3.3 respectively denote the minimum, maximum and average lengths of mesh edges.

Mesh	**# elements**	$\mathbf{L_{min}}$ **(mm)**	$\mathbf{L_{max}}$ **(mm)**	$\mathbf{L_{avg}}$ **(mm)**
M1	815,405	1.00	28.14	10.69
M2	1,862,136	0.65	23.81	6.89
M3	7,894,112	0.77	22.75	3.21

TABLE 4.3.3 – Characteristics of the fully unstructured tetrahedral meshes of head tissues.

Fig. 4.5 shows the contour lines of the amplitude of the electric field on the skin, skull and brain surfaces for a numerical solution computed with mesh M2 using the DGTD-$\mathbb{P}_2$ method. Contour lines of the local SAR normalized to the maximum value of the local SAR and of the local SAR normalized to the total emitted power, plotted in selected XoZ and XoY planes, are shown on Fig. 4.6 and Fig. 4.7 for calculations based on the coarsest mesh (*i.e.* mesh M1), and on Fig. 4.8 and Fig. 4.9 for calculations

based on the finest mesh (*i.e.* mesh M3). In order to discuss these results, we consider that the numerical solution computed with mesh M3 using the DGTD-$\mathbb{P}_1$ method defines a reference solution. Patterns of the contour lines for calculations respectively performed with mesh M1 using the DGTD-$\mathbb{P}_3$ method and with mesh M3 using the DGTD-$\mathbb{P}_1$ method are very similar. However variations certainly exist locally since the ranges of the plotted values differ (especially for the local SAR normalized to the total emitted power).

In Table 4.3.4, the quantity given parenthetically is the difference with the reference value (*i.e.* , the one associated to mesh M3 and the DGTD-$\mathbb{P}_1$ method) and the corresponding error level. We note here that the relative error tends to increase when switching from $p = 1$ to $p = 2$ for a given discretization mesh. Again, this should not be interpreted as a counter effect of an increase of the approximation order since this relative error is evaluated on the basis of a solution computed on the finest mesh which is constructed from high resolution triangulations of the tissue interfaces (see the figures in section 4.3.3.2). Rather, we can conclude that the discretization of the geometrical features of tissues has a greater impact on accuracy than the interpolation order in the DGTD-$\mathbb{P}_p$ method (first column of Table 4.3.3).

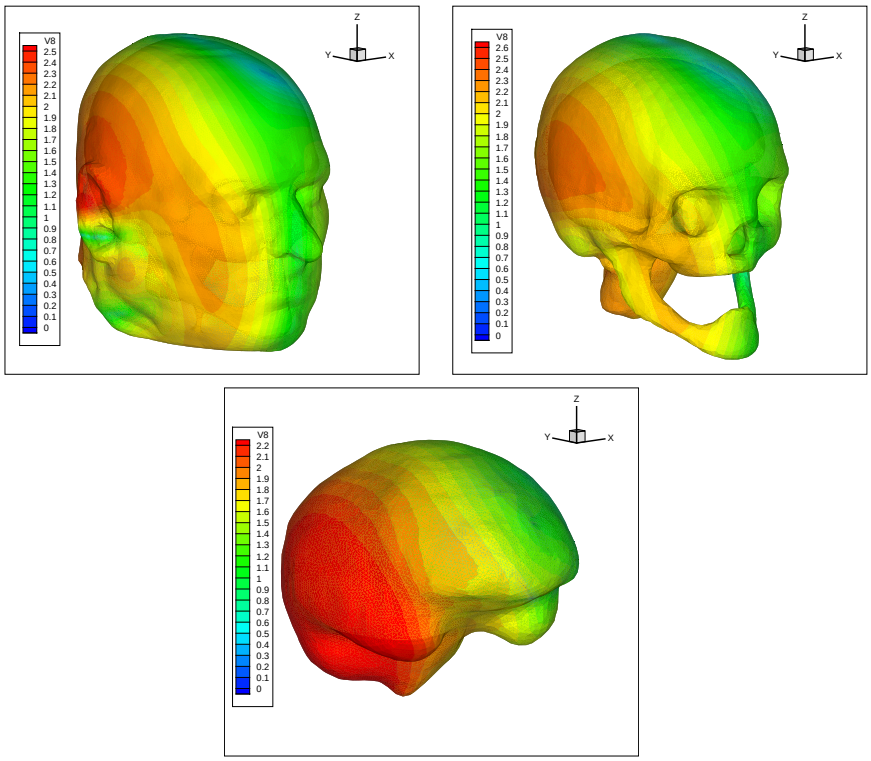


Figure 4.5 – Calculations with the DGTD-$\mathbb{P}_2$ method. Mesh M2 : contour lines of the amplitude of the electric field in log scale.

4.3.3.4 Performance results

We now come back to the wave propagation problems discussed in subsection 4.3.3.3 and present strong scalability results for simulations with the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods using the tetrahedral meshes of Table 4.3.3. Performances results are presented in Tables 4.3.5 to 4.3.7. For the coarsest mesh (*i.e.* mesh M1), the parallel speedup is evaluated for 16 GPUs relatively to the simulation time using one GPU. Although the number of elements of this mesh is well below the size of the mesh considered for the weak scalability analysis (*i.e.* 3,072,000 elements for the DGTD-$\mathbb{P}_1$ and DGTD-$\mathbb{P}_2$ methods), superlinear speedups are obtained. However, not surprisingly, the single GPU GFlops rates are lower

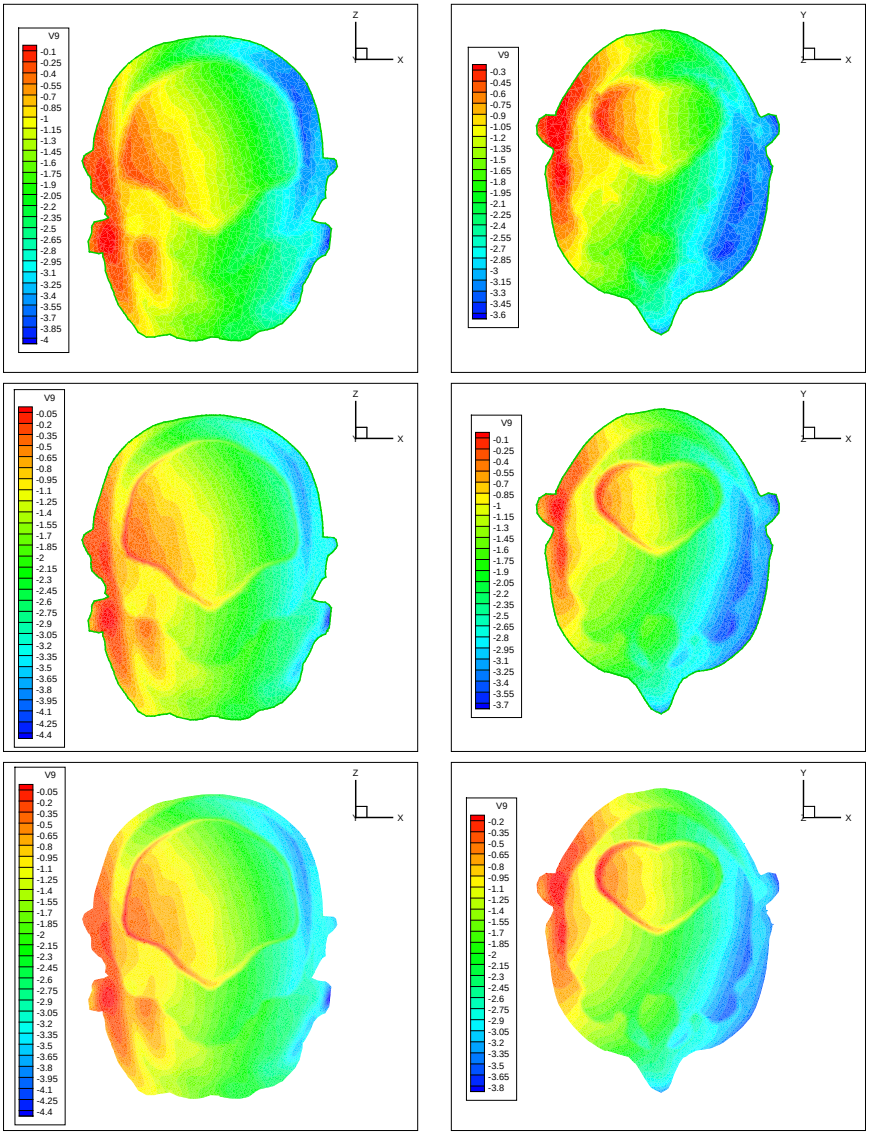


Figure 4.6 – Calculations with the DGTD-$\mathbb{P}_1$ (top), DGTD-$\mathbb{P}_2$ (middle) and DGTD-$\mathbb{P}_3$ (bottom) methods. Mesh M1 : contour lines of local SAR over maximum local SAR in log scale (selected cut planes).

Mesh	Method
-	DGTD-$\mathbb{P}_1$
M1	3.365 W/Kg (0.463, 12.1 %)
M2	3.734 W/Kg (0.094, 2.4 %)
M3	3.828 W/Kg (reference value)
-	DGTD-$\mathbb{P}_2$
M1	3.269 W/Kg (0.559, 14.6 %)
M2	3.586 W/Kg (0.242, 6.3 %)
-	DGTD-$\mathbb{P}_3$
M1	3.283 W/Kg (0.545, 12.3 %)

Table 4.3.4 – Calculations with the DGTD-$\mathbb{P}_p$ method. Maximum value of the normalized local SAR.

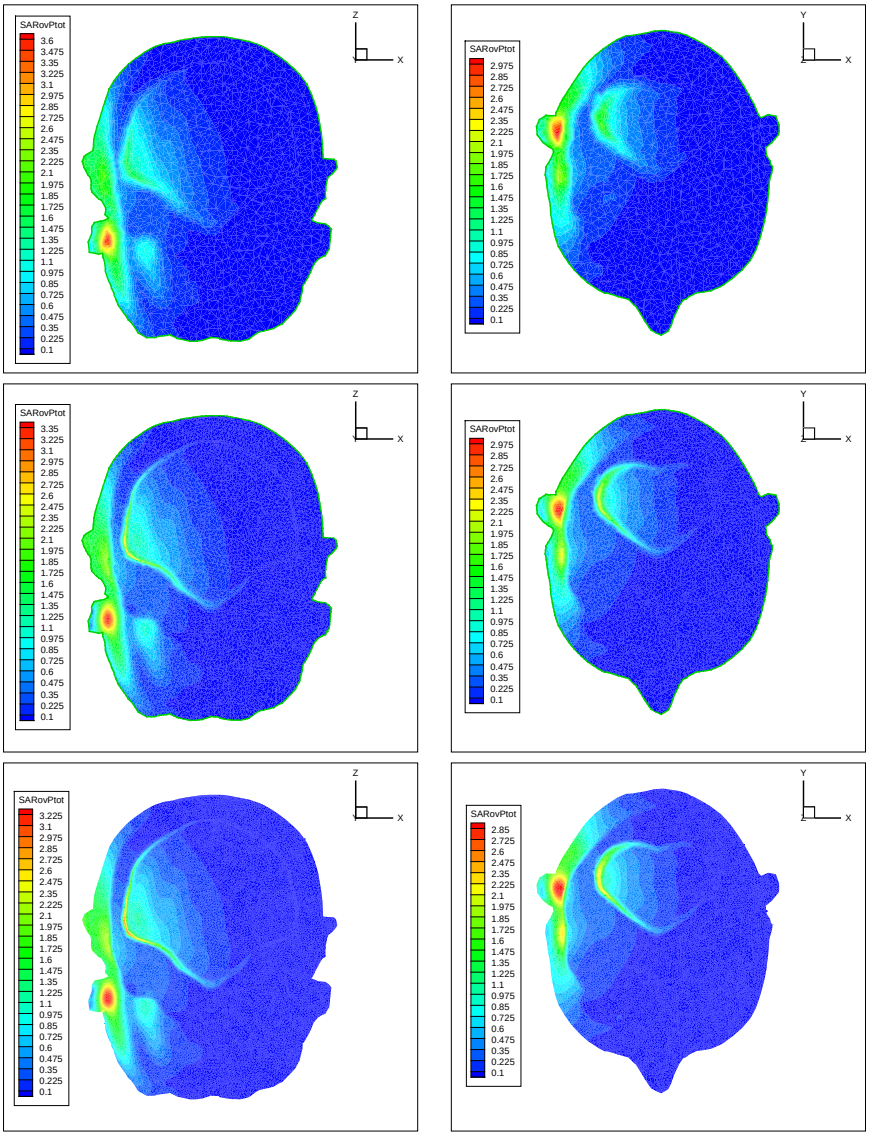


Figure 4.7 – Calculations with the DGTD-$\mathbb{P}_1$ (top), DGTD-$\mathbb{P}_2$ (middle) and DGTD-$\mathbb{P}_3$ (bottom) methods. Mesh M1 : contour lines of normalized local SAR (selected cut planes).

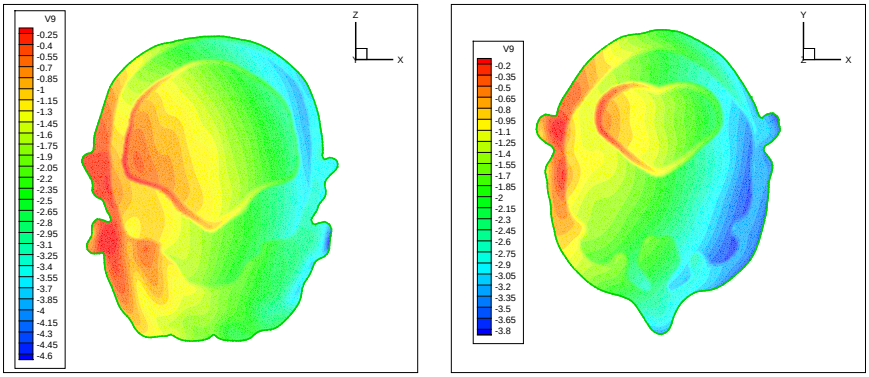


Figure 4.8 – Calculations with the DGTD-$\mathbb{P}_1$ method. Mesh M3 : contour lines of local SAR over maximum local SAR in log scale (selected cut planes).

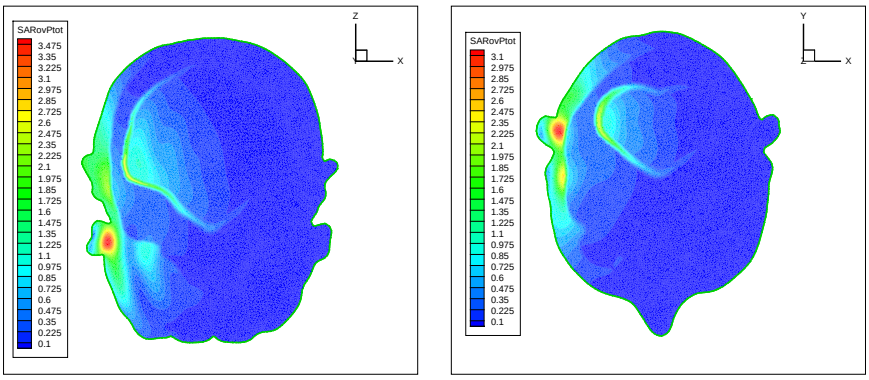


Figure 4.9 – Calculations with the DGTD-$\mathbb{P}_1$ method. Mesh M3 : contour lines of normalized local SAR (selected cut planes).

than the corresponding ones reported in Table 4.3.1 (32 instead of 63 for the DGTD-$\mathbb{P}_1$ method, and 60 instead of 92 for the DGTD-$\mathbb{P}_2$ method). For the two other meshes (*i.e.* M2 and M3), as expected the DGTD-$\mathbb{P}_2$ method is always more scalable than the DGTD-$\mathbb{P}_1$ method because of a more favorable computation to communication ratio. Overall, acceleration factors ranging from 20 to 26 for the mesh M1, from 15 to 24 for the mesh M2 and from 24 to 40 for the mesh M3 are observed between the multiple CPU and multiple GPU simulations. We note however that this comparison is made with a CPU version whose parallel implementation relies on MPI only. In particular, we have not considered a possible optimization to hybrid shared-memory multi-core systems combining the OpenMP and MPI programming models. Besides, an optimized CPU version in terms of simulation times can be obtained by computing the surface integrals over $\partial\tau_i$ in (4.2.1) through a loop over element faces and updating the flux balance of both elements τ_i and τ_j since the numerical flux between τ_j and τ_i is just the opposite of that from τ_i and τ_j. Such an optimization would lower the simulation times of the CPU version by approximately 30%. In the present implementation, each elementary numerical flux is computed twice (respectively for flux balances of τ_i and τ_j) for maximizing the floating point performance in the CUDA SIMD framework.

# GPU	DGTD-$\mathbb{P}_1$			DGTD-$\mathbb{P}_2$		
	Time	GFlops	Speedup	Time	GFlops	Speedup
1	620 sec	32	-	2683 sec	60	-
16	35 sec	566	17.8	145 sec	1110	18.5

Table 4.3.5 – Head tissues exposure to an electromagnetic wave emitted by a mobile phone. Strong scalability assessment : mesh M1. Elapsed time on 16 CPUs : 715 sec (DGTD-$\mathbb{P}_1$ method) and 3824 sec (DGTD-$\mathbb{P}_2$ method).

# GPU	DGTD-$\mathbb{P}_1$			DGTD-$\mathbb{P}_2$		
	Time	GFlops	Speedup	Time	GFlops	Speedup
16	82 sec	699	-	407 sec	1137	-
32	46 sec	1239	1.8	201 sec	2299	2.0
64	33 sec	1747	2.5	116 sec	4007	3.5

Table 4.3.6 – Head tissues exposure to an electromagnetic wave emitted by a mobile phone. Strong scalability assessment : mesh M2. Elapsed time on 64 CPUs : 519 sec (DGTD-$\mathbb{P}_1$ method) and 2869 sec (DGTD-$\mathbb{P}_2$ method).

# GPU	DGTD-$\mathbb{P}_1$			DGTD-$\mathbb{P}_2$		
	Time	GFlops	Speedup	Time	GFlops	Speedup
32	162 sec	1460	-	816 sec	2370	-
64	97 sec	2470	1.7	416 sec	4657	2.0
128	69 sec	3469	2.4	257 sec	7522	3.2

TABLE 4.3.7 – Head tissues exposure to an electromagnetic wave emitted by a mobile phone. Strong scalability assessment : mesh M3. Elapsed time on 64 CPUs : 2786 sec (DGTD-$\mathbb{P}_1$ method) and 6057 sec (DGTD-$\mathbb{P}_2$ method).

4.4 Extension to the Fermi architecture

Based on previous considerations, we discuss in this section an approach to parallelizing the DGTD-$\mathbb{P}_{p_i}$ method on the Fermi architecture within CUDA 4.0 to get the most out of our many-core investment. As we exposed in A.1.1.3 of annex A, the Fermi architecture introduced many anticipated improvements that affect, and facilitate programming on NVIDIA GPUs. Besides, porting of the simulation software will take advantage of CUDA 4.0's specific features (such as P2P or Unified Virtual Addressing) in the framework of multi-GPU programming using CUDA C, MPI and NVIDIA's GPUDirect 2.0.

In the previous section, substantial peak computing power in hybrid CPU-GPU clusters was observed when using multiple GPUs in each compute node by interconnecting the nodes with a fast network. To that end, we considered a MPI-CUDA implementation on multi-GPU clusters with a high-end GT200 GPU per compute-node, running at 102.4 GB/s and processing theoretically 933 GFlops and 77 GFlops in single and double precision floating point arithmetic respectively. In this section, we investigate the adaptation and tuning to Fermi architecture of the existing GPU-enabled DGTD method discussed in 4.2.1. Numerical experiments will be performed on a Fermi-based hybrid CPU-GPU cluster localized at the INRIA Bordeaux - Sud-Ouest research center, accommodating 24 Tesla M2070 GPUs in total, with 3 GPUs per node. Simulations are performed using single precision floating point as double precision calculations usually provides reduced performances and half of the memory storage.

4.4.1 MPI-CUDA implementation

As discussed in subsection 4.2.2, the tetrahedral mesh we consider to fully discretize a 3D domain is decomposed into submeshes. In this implementation, each MPI process is assigned a unique GPU identifier and we generate within each MPI process a unique set of data related to a submesh. MPI data transfers across the neighbouring GPUs must be performed carefully and efficiently to achieve high throughput and scalable results ; while the number of GPUs is increased, data travelling across the network will increase and inject additional latency into the implementation, which can restrict scalability if not properly handled. The multi-GPU parallelization strategy we adopted requires that we compute intVolume and exchange data between MPI processes prior to compute intSurface, hence we use non-blocking MPI communications and asynchronous memory transfers to overlap local GPU computations of the volume integrals in 4.2.3 and cudaHostAlloc buffers to let the driver handle CPU-GPU copies (see Fig. 4.10).

4.4.2 Hardware configuration

PLAFRIM, the hybrid CPU-GPU cluster localized at the INRIA Bordeaux - Sud-Ouest research center, consists of 8 Intel CPU nodes connected by an InfiniBand QDR (Quad Data Rate) network with a 40 Gb/s speed. Each compute node has two hexa-core 2.67 GHz Wetsmare Intel Xeon X5650 processors and 36 Go of host memory. The cluster is comprised of 24 NVIDIA Tesla M2070 GPU systems with 3 GPUs available per node. The Tesla M2070 GPU, with 448 CUDA cores, delivers 1.288 TFlops of peak

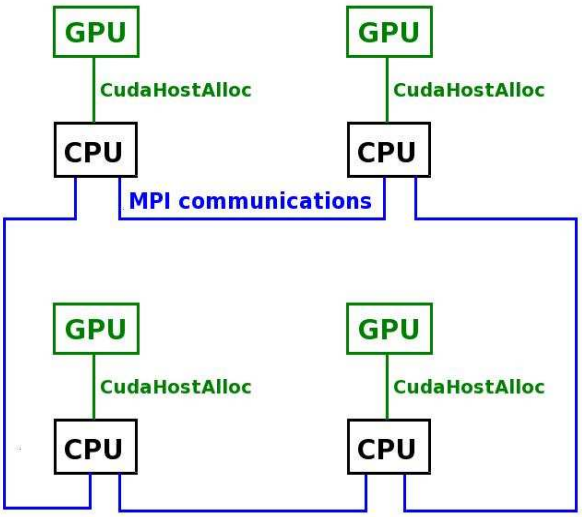


Figure 4.10 – MPI-CUDA implementation.

single precision floating point performance (515 GFlops in double precision), 150.3 GB/s of maximum bandwidth and includes 6 GB of memory.

To discuss the capabilities of heterogeneous simulations on multi-GPU clusters, we first analyse and maximize the performance behaviour of our code on a single Tesla M2070 GPU. Then we conduct weak and strong scaling experiments to assess the runtime performance and efficiency of the MPI-CUDA implementation.

4.4.3 Fermi adaptation : kernel improvements

With the appearance of Fermi's new streaming multiprocessors (SMs) and the launch of CUDA 4.0 GPU computing environment, NVIDIA provided software developers a boost of performance and extended features that take full advantage of GPU acceleration. We enhanced our MPI-CUDA implementation by applying low-level optimizations to the GPU kernels. First we took advantage of the improved memory subsystem offering a unified L2 cache and a L1 cache per SM, and made the choice of a bigger shared memory using the « cudaThreadSetCacheConfig(cudaFuncCachePreferShared) »function. Then we improved single GPU performance giving each thread two degrees of freedom to compute, instead of one (see section A.3 of annex A), for most of the kernels. Combined with larger thread block sizes, it gives a bigger amount of data parallel work per clock cycle and enables CUDA kernels to run more efficiently on large parallel machines. Finally, relying on a CUDA programming guide advice justified by the advent of the coherent Fermi L2 cache (see [NVIDIA 2011]), we attempted the use of global memory instead of texture memory but it turned out that performance levels were unfortunately not conclusive.

4.4.4 Performance results on a model test problem

We first give performance results for the model problem of the propagation of an eigenmode in a cubic cavity, involving a mesh consisted of 384,000 elements to illustrate the resulted performance gains. We estimate the sequential computing time for the simulation of the DGTD-$\mathbb{P}_2$ method for a duration of 1000 time steps and summarize the corresponding results in Table 4.4.8, showing a significant reduction of the computational cost.

We now analyze the weak scaling on the Plafrim cluster. Using box-wise domain decompositions with optimal computational load balance, we make sure that the work load per compute node is kept constant

Kernel	GT200		Fermi		Gain %
	Gbytes/s	GFlops	Gbytes/s	GFlops	
intVolume	67.9	270	75	298	10
intSurface	199	132	205	137	4
updateE	131	117	179	160	37
updateH	133	100	162	122	22
updateH+enrj	96	111	137	158	42

TABLE 4.4.8 – Fermi adaptation : kernel improvements

for an increasing number of CPU cores. The problem size per processor remains therefore unchanged during the mesh refinement process. We recall that the problem size for one GPU represents $3,072,000$ tetrahedral elements for the DGTD-$\mathbb{P}_1$ ($73,728,000$ d.o.f.) and DGTD-$\mathbb{P}_2$ methods ($184,320,000$ d.o.f.), $1,296,000$ elements for the DGTD-$\mathbb{P}_3$ method ($155,520,000$ d.o.f.) and $750,000$ elements for the DGTD-$\mathbb{P}_4$ method ($157,500,000$ d.o.f.). Due to hardware restrictions, we perform weak scaling experiments using up to 16 GPUs on 8 compute nodes as only 24 GPU nodes are available. Results are depicted on Fig. 4.11 and show a very good parallel efficiency of the GPU enabled DGTD-$\mathbb{P}_{p_i}$ method considered in subsection 3.2.2. Time measurements correspond to a period of 1000 iterations of the parallelized loop involving the leap-frog time scheme (1.2.16). GFlops rates are represented on Fig. 4.12 and compared to the performance results obtained in subsection 4.3.2 for the GT200 GPU-enabled method. Not surprisingly, improved performance and efficiency are observed for our fixed-size problem when using tuned Fermi kernels (see Tables 4.4.9 and 4.4.10 for more details).

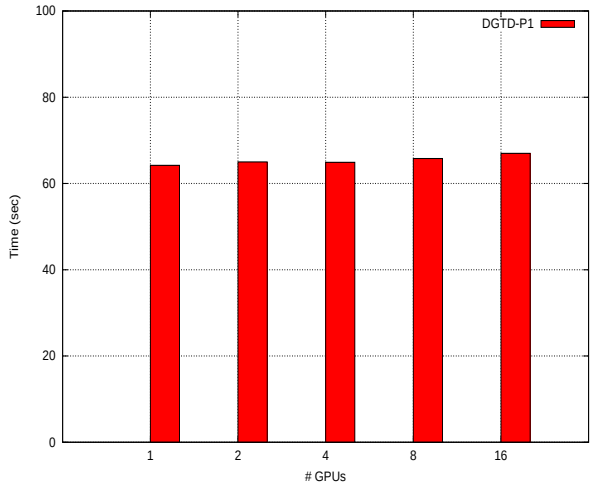
# GPU	DGTD-$\mathbb{P}_1$		DGTD-$\mathbb{P}_2$		DGTD-$\mathbb{P}_3$		DGTD-$\mathbb{P}_4$	
	Time	GFlops	Time	GFlops	Time	GFlops	Time	GFlops
1	104.7 sec	63	325.1 sec	92	410.3 sec	106	759.8 sec	94
128	104.9 sec	8072	323.1 sec	11844	408.4 sec	13676	763.6 sec	12009

TABLE 4.4.9 – Weak scalability assessment : timings and sustained performance figures (C1060).

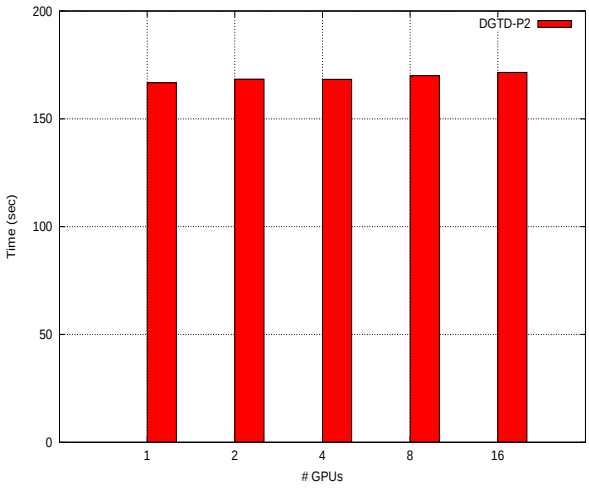
# GPU	DGTD-$\mathbb{P}_1$		DGTD-$\mathbb{P}_2$		DGTD-$\mathbb{P}_3$		DGTD-$\mathbb{P}_4$	
	Time	GFlops	Time	GFlops	Time	GFlops	Time	GFlops
1	64.2 sec	106	166.7 sec	181	194.0 sec	226	370.5 sec	195
16	67.0 sec	1620	171.5 sec	2820	198.2 sec	3540	381.9 sec	3025

TABLE 4.4.10 – Weak scalability assessment : timings and sustained performance figures (M2070).

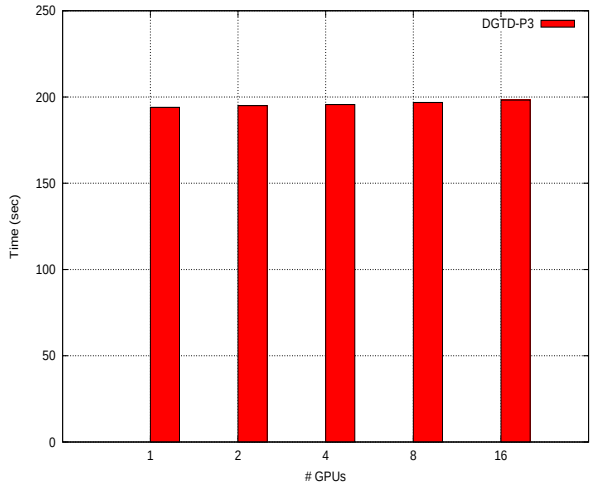
In order to have a better insight of the performance increase between the two generations of NVIDIA GPUs, we consider the same test problem using a smaller mesh containing $384,000$ elements and successively analyse speedups for sequential runs on 4 different NVIDIA Fermi graphic cards (see Table 4.4.11) versus the previously used GT200 GPU. We perform 1000 iterations of the leap-frog time scheme in each run. As PCIe bus traffic suffers from bandwidth congestion, we minimize data transfers between the GPU and the CPU host by allocating and loading local and global arrays on the GPU in the initialization step, prior to the start of the time loop, in the limit of the available amount of memory in each GPU. Performance measurements and speedups of Fermi GPUs versus the GT200 GPU are presented respectively in Tables 4.4.12 and 4.4.13. As one may observe, processing on different architectures families clearly impact on GPU calculations. As expected, usage of different GPUs of the same Tesla family makes no significant change in performance, nonetheless considering porting the application is relevant



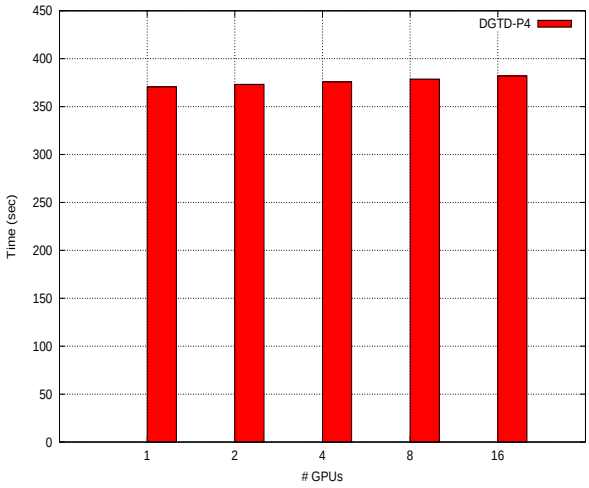
DGTD-$\mathbb{P}_1$ method



DGTD-$\mathbb{P}_2$ method

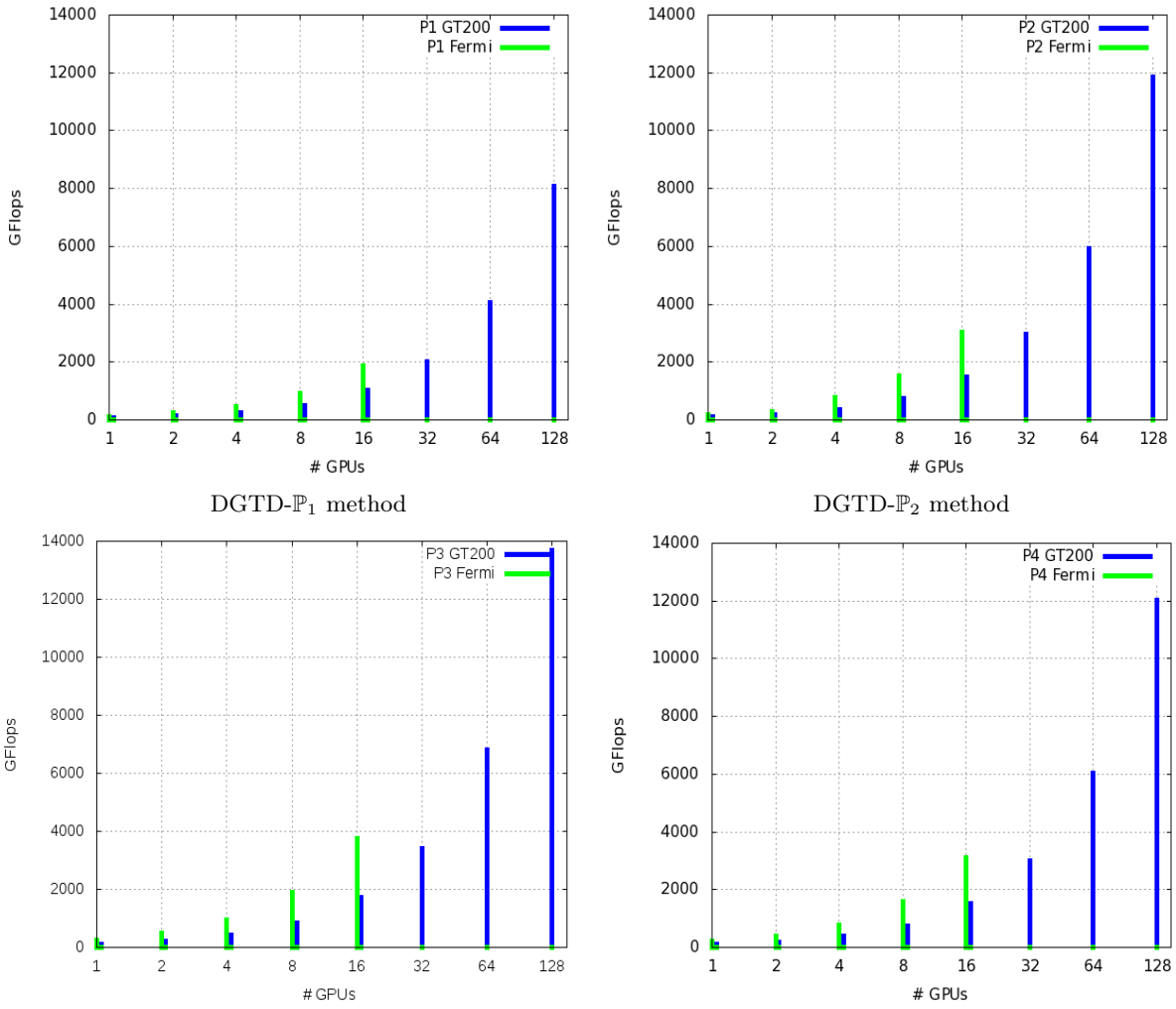


DGTD-$\mathbb{P}_3$ method



DGTD-$\mathbb{P}_4$ method

Figure 4.11 – Weak scalability assessment : evolution of the computing time with the number of GPUs.



DGTD-$\mathbb{P}_1$ method DGTD-$\mathbb{P}_2$ method

DGTD-$\mathbb{P}_3$ method DGTD-$\mathbb{P}_4$ method

Figure 4.12 – Weak scalability assessment : evolution of GFlops rates with the number of GPUs.

to ensure correctness of the code and to enable maximum reliability of algorithm performances. Another feature worth mentioning here is the net gain in performance of a factor of two for the DGTD-$\mathbb{P}_3$ method, followed by a drop in speedup for the DGTD-$\mathbb{P}_4$. This loss of performance is caused by a suboptimal usage of memory bandwidth characterized by an overload of computations being performed beyond the third order scheme.

GPU	Theoretical values	
	GBytes/s	GFlops
C1060	102.4	933
M2050	148.4	1288
C2050	144.0	1288
C2070	144.0	1288
M2070	150.3	1288

TABLE 4.4.11 – Peak single precision floating point performance and maximum bandwidth of 4 different NVIDIA Fermi graphics cards relative to the performance of the GT200 implementation.

GPU	DGTD-$\mathbb{P}_1$		DGTD-$\mathbb{P}_2$		DGTD-$\mathbb{P}_3$		DGTD-$\mathbb{P}_4$	
	Time	GFlops	Time	GFlops	Time	GFlops	Time	GFlops
C1060	12.2 sec	70	34.2 sec	111	109.4 sec	119	330.7 sec	112
M2050	7.0 sec	120	19.3 sec	195	53.8 sec	241	182.3 sec	203
C2050	7.1 sec	119	19.5 sec	194	54.4 sec	239	183.2 sec	202
C2070	7.3 sec	117	19.9 sec	190	55.0 sec	236	186.0 sec	199
M2070	7.1 sec	119	19.6 sec	193	54.5 sec	239	184.0 sec	201

TABLE 4.4.12 – Time loop performances for sequential runs - Mode (nk, nk, nk) in cubic cavity - $41 \times 41 \times 41$ (384, 000 elements) - 1000 iterations - Fermi vs GT200

4.4.5 Performance results on a realistic problem

Multi-GPU acceleration of the DGTD method using GT200 GPUs has previously been illustrated through the simulation of human head tissues exposure to an electromagnetic wave emitted by a mobile phone (see section 4.3.3 for a complete definition of this problem) Although our related CUDA implementation would certainly benefit from further optimizations such as a better GPU memory occupation in view of reducing the overall network traffic, strong scalability assessments clearly exhibited a significant gain of performance. Referring to Table 4.3.7, we recall that for the simulations involving a large problem size (mesh M3 of Table 4.3.3 consisting of $7,894,112$ elements), using a total of 128 GPUs deployed on 128 compute nodes of the CCRT cluster, we managed to sustain a performance of 7,5 TFlops.

In this part, the methodology is extended to NVIDIA Fermi architecture and a different problem size is considered to investigate the strong scalability of the corresponding DGTD implementation. As a smaller number of Fermi GPUs is available in the PLAFRIM cluster, strong scaling experiments are successively

GPU	Speedups			
	DGTD-$\mathbb{P}_1$	**DGTD-$\mathbb{P}_2$**	**DGTD-$\mathbb{P}_3$**	**DGTD-$\mathbb{P}_4$**
C1060	-	-	-	-
M2050	×1.74	×1.77	×2.03	×1.81
C2050	×1.72	×1.75	×2.01	×1.81
C2070	×1.67	×1.72	×1.99	×1.78
M2070	×1.72	×1.74	×2.01	×1.80

TABLE 4.4.13 – Speedups for sequential runs - Mode (nk, nk, nk) in cubic cavity - $41 \times 41 \times 41$ (384, 000 elements) - ktmax = 1000 - Fermi vs GT200

performed on 4, 8 and 16 accelerating graphic cards. As we have a set of 3 GPUs per compute node at our disposal in the PLAFRIM cluster, it is worth mentioning that processes within a same node will take advantage of a high speed inter-node memory communication, thus improving data sharing performances. Our Fermi implementation of the DGTD method is applied here to the simulation of the scattering of a plane wave by an aircraft geometry (see Fig. 4.13), involving $2,024,924$ elements, *i.e.* handling $48,598,176$ d.o.f. for the DGTD-$\mathbb{P}_1$ method, $121,495,440$ d.o.f. for the DGTD-$\mathbb{P}_2$ method, $242,990,880$ d.o.f. for the DGTD-$\mathbb{P}_3$ method and $425,234,040$ d.o.f. for the DGTD-$\mathbb{P}_4$ method. Performance timings are given for 1000 iterations of a time stepping loop. Performance figures are given in Tables 4.4.14 and 4.4.15. We note that, with our Fermi implementation, on can reach a near-linear speedup with the DGTD-$\mathbb{P}_4$ method whereas such a performance is not achieved when using lower interpolation orders.

Finally, Tables 4.4.16 to 4.4.20 give a more detailed analysis of the parallel efficiency and optimization potential of the Fermi multi-GPUs implementation with regard to a simulation partitioned across multiple compute nodes using Fortran with embedded MPI calls. Additionally, results are compared to performances obtained with a hybrid GPU-based system consisting of Tesla C2050 and C2070 accelerating graphic cards, and composed of two compute nodes, each filled with seven identical GPUs. The considered application, distributed on 4 NVIDIA M2070 GPUs, delivers substantial speedups (between ×33 and ×56) over multi-core CPUs but displays a less important performance increase when evaluated on NVIDIA Tesla C2070 GPU cards (between ×23 and ×51). Besides, parallel efficiency drops down when the simulation is configured to run on more than 4 GPUs, yielding lower speedups over the CPU solution. Even though our Maxwell solver only achieved a sustained performance of 440 GFlops (*i.e.* 34,2% of the theoretical peak GPU performance) on a single node, scaling and speedup results on the PLAFRIM cluster are very promising over previous GPU algorithms and we expect this overall computational speedup to be maintained on larger clusters enabling large-scale parallelism.

# GPU	DGTD-$\mathbb{P}_1$			DGTD-$\mathbb{P}_2$		
	Time	**GFlops**	**Speedup**	**Time**	**GFlops**	**Speedup**
4	18.6 sec	242	-	42.7 sec	468	-
8	12.6 sec	355	×1.5	25.1 sec	795	×1.7
16	9.7 sec	464	×1.9	16.1 sec	1239	×2.7

TABLE 4.4.14 – Strong scalability (Fermi M2070), 1000 iterations.

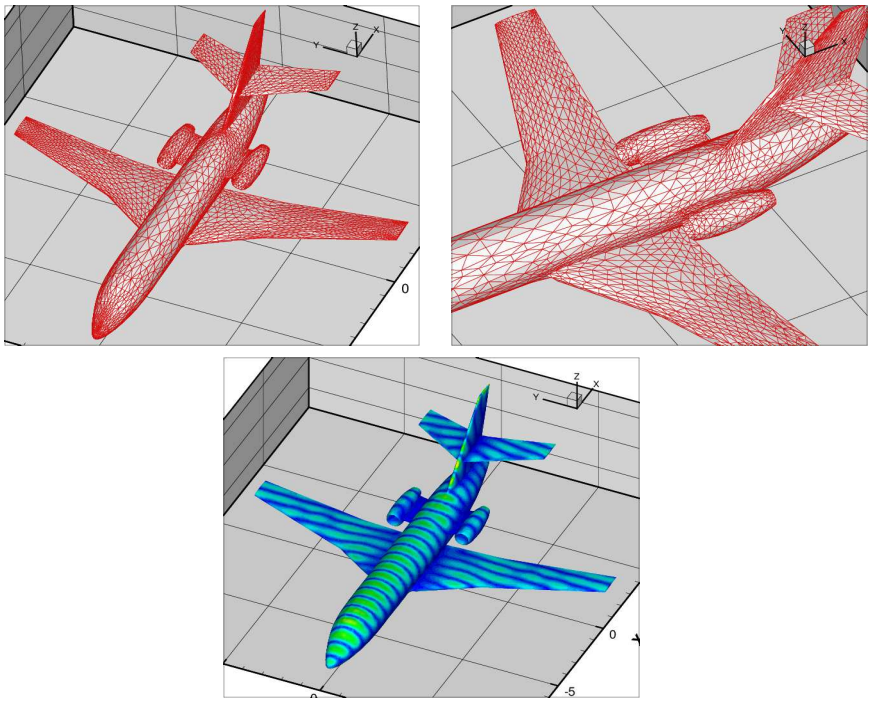


Figure 4.13 – Calculations with the DGTD-$\mathbb{P}_2$ method. Contour lines of the amplitude of the electric field in log scale.

4.5 Conclusion

In this chapter, we have presented a high performance numerical methodology to simulate electromagnetic wave propagation in complex domains and heterogeneous media. This methodology is based on a high order discontinuous Galerkin time domain method formulated on unstructured tetrahedral meshes for solving the system of Maxwell equations. Due to its intrinsically local nature, this DGTD method is particularly well suited to distributed memory parallel computing. Besides, from the algorithmic point of view, the method mixes sparse linear algebra operations (as usual with classical finite element or finite volume methods) with dense linear algebra operations due to the use of a high order nodal interpolation method at the element level. Therefore, the method is an ideal candidate for exploiting the processing capabilities of GPU systems. In this work, this DGTD method has been adapted to multi-GPU parallel computing by combining a coarse grain SPMD programming model for inter-GPU parallelization and a fine grain SIMD programming model for intra-GPU parallelization. The methodology has been applied in this paper to the numerical evaluation of the SAR induced in head tissues when the latter are exposed to an electromagnetic wave emitted by a mobile phone antenna. The underlying electromagnetic wave propagation problems are particularly challenging because of the heterogeneity of the propagation media

# GPU	DGTD-$\mathbb{P}_3$			DGTD-$\mathbb{P}_4$		
	Time	GFlops	Speedup	Time	GFlops	Speedup
4	89.0 sec	771	-	265.6 sec	735	-
8	48.5 sec	1414	×1.8	139.7 sec	1396	×1.9
16	28.0 sec	2452	×3.2	75.2 sec	2593	×3.5

Table 4.4.15 – Strong scalability (Fermi M2070), 1000 iterations.

GPU	DGTD-$\mathbb{P}_1$		DGTD-$\mathbb{P}_2$		DGTD-$\mathbb{P}_3$		DGTD-$\mathbb{P}_4$	
	Time	GFlops	Time	GFlops	Time	GFlops	Time	GFlops
CPU	628.9 sec	-	1778.3 sec	-	4995.0 sec	-	11439.8 sec	-
C2050	20.3 sec	221	43.9 sec	455	90.1 sec	761	265.8 sec	734
C2070	26.8 sec	168	51.5 sec	388	98.2 sec	698	278.5 sec	701
M2070	18.6 sec	242	42.7 sec	468	89.0 sec	771	265.6 sec	735

TABLE 4.4.16 – Detailed performance figures on 4 GPUs, 1000 iterations.

GPU	DGTD-$\mathbb{P}_1$		DGTD-$\mathbb{P}_2$		DGTD-$\mathbb{P}_3$		DGTD-$\mathbb{P}_4$	
	Time	GFlops	Time	GFlops	Time	GFlops	Time	GFlops
CPU	324.0 sec	-	902.0 sec	-	2517.4 sec	-	5736.3 sec	-
C2050/C2070	20.1 sec	223	32.6 sec	612	53.6 sec	1218	148.4 sec	1315
M2070	12.6 sec	355	25.1 sec	795	48.5 sec	1414	139.7 sec	1396

TABLE 4.4.17 – Detailed performance figures on 8 GPUs, 1000 iterations.

GPU	DGTD-$\mathbb{P}_1$		DGTD-$\mathbb{P}_2$		DGTD-$\mathbb{P}_3$		DGTD-$\mathbb{P}_4$	
	Time	GFlops	Time	GFlops	Time	GFlops	Time	GFlops
C2050/C2070	21.3 sec	211	32.5 sec	614	56.5 sec	1214	151.2 sec	1290
M2070	9.7 sec	464	16.1 sec	1239	28.0 sec	2452	75.2 sec	2593

TABLE 4.4.18 – Detailed performance figures on 16 GPUs, 1000 iterations.

GPU	Speedups			
	DGTD-$\mathbb{P}_1$	DGTD-$\mathbb{P}_2$	DGTD-$\mathbb{P}_3$	DGTD-$\mathbb{P}_4$
C2050	×31.0	×40.5	×55.4	×43.0
C2070	×23.5	×34.5	×50.9	×41.1
M2070	×33.8	×41.6	×56.1	×43.1

TABLE 4.4.19 – Performance comparison 4 GPUs, 1000 iterations.

GPU	Speedups			
	DGTD-$\mathbb{P}_1$	DGTD-$\mathbb{P}_2$	DGTD-$\mathbb{P}_3$	DGTD-$\mathbb{P}_4$
C2050/C2070	×16.1	×27.7	×47.0	×38.6
M2070	×25.7	×35.9	×51.9	×41.1

TABLE 4.4.20 – Performance comparison 8 GPUs, 1000 iterations.

and the complexity of the shapes of the head tissues. Unstructured mesh based solvers are attractive in this context since they allow for an accurate discretization of the interfaces between tissues where discontinuities of the field components occur. Noteworthy, a complete picture of the assessment of the potential adverse effects of head tissues exposure to mobile phone radiation requires to consider several factors such as the variations of the morphology and the electromagnetic characteristics of the tissues with the age, as well as the the type and positioning of the source (*i.e.* the mobile phone antenna). Whether such a study is conducted by a multi-parametric analysis or by relying on a strategy for uncertainty analysis, reducing the computational time of a single 3D simulation is highly desirable if not mandatory. We have then ported the high order DGTD method on a cluster of Fermi GPUs to further speed up calculations. Likewise, we combined GPU CUDA programming with non-blocking message passing based on MPI to overlap the communications across the network and the data transfer to and from the device via PCIe with calculations on the GPU. With the resulting methodology applied to the simulation of the scattering of a plane wave by an aircraft geometry, we performed several numerical tests to compare the performance between the MPI-CUDA implementation on latest GPU hardware with respect to the original version of the code without CUDA. We obtained speedup factors ranging between 16× and 56× on multiple GPUs with respect to the CPU solution, depending on the interpolation order considered as well as on the number and family of accelerating graphic Fermi cards used. As expected, performance measurements also achieved a performance increase close to 2× when turning to NVIDIA Fermi architecture compared to the GT200 implementation. Thus, the multi-GPU parallelization strategy we adopted turned out to be a meaningful and very worthwhile objective. However the performance boost would certainly benefit from further in-depth optimizations of kernels, maximizing thread occupancy to better hide latencies and enhance the achieved peak performance.

Acknowledgments. This work was granted access to the HPC resources of CCRT under the allocation 2010-t2010065004 made by GENCI (Grand Equipement National de Calcul Intensif) and to the HPC resources of the experimental platform Plafrim shared between IMB (Institut Mathématiques de Bordeaux), LaBRI (Laboratoire Bordelais de Recherche en Informatique) and INRIA Bordeaux - Sud-Ouest (Institut National de Recherche en Informatique et en Automatique).

Chapitre 5

Conclusion générale

Les contributions proposées dans cette thèse s'inscrivent dans une initiative plus globale qui vise à la mise au point d'une méthodologique numérique précise, flexible (*hp*-adaptative) et performante (adaptée au calcul haute performance) pour la simulation de problèmes de propagation d'ondes électromagnétiques tridimensionnels en régime temporel, mettant en jeu des milieux de propagation complexes et des structures géométriques irrégulières. Les résultats obtenus à ce stade sont encourageants mais il est clair que ces développements en appellent d'autres pour atteindre l'objectif escompté. En particulier, dans le prolongement de nos travaux, les points suivants devraient être abordés :

- Sur les bases de fonctions polynomiales modales hiérarchiques telles que la base de Solin, le caractère hiérarchique doit être pleinement exploité du point de vue algorithmique pour permettre une mise en œuvre flexible de la *p*-adaptivité i.e. la possibilité d'adapter dynamiquement le degré d'interpolation au sein de chaque élément du maillage au cours d'une simulation. A noter qu'aucune des bases considérées dans la présente étude pour la mise au point d'une méthode GDDT-$\mathbb{P}_p$ en maillages simplexes n'est orthogonale. Il semble que des travaux récents (qui n'ont pu être considérés ici par faute de temps) ont conduit à la mise au point de bases hiérarchiques orthogonales sur des simplexes en 2D et 3D [Xin 2012]. Il serait intéressant de compléter notre étude avec la prise en compte de cette base en triangle, l'avantage certain de l'orthogonalité étant le gain en temps de calcul : pas d'inversion de la matrice de masse locale et calcul immédiat des produits matrice-vecteur avec la matrice de masse et son inverse. Néanmoins, il faudra probablement modérer ce gain par une complexité de programmation accrue (comme observée ici pour les bases hiérachiques de Solin, Ainsworth-Coyle et Sherkin-Karniadakis). Un objectif à plus long terme, en continuité avec les travaux de cette thèse, est la construction et l'étude comparative d'estimateurs d'erreur a posteriori pour l'approximation des équations de Maxwell en domaine temporel en vue d'obtenir une méthodologie GDDT-$\mathbb{P}_p$ *hp*-adaptative pour la simulation numérique de problèmes de propagation tridimensionnels.
- L'utilisation d'un schéma d'intégration en temps explicite conditionnellement stable sur un maillage localement raffiné suppose, par le biais de la condition CFL, que le pas de temps global est fixé par la taille du plus petit élément. Ainsi, la modélisation numérique de problèmes de propagation d'ondes électromagnétiques faisant intervenir des géométries complexes ou nécessitant la discrétisation de milieux de propagation hétérogènes, de matériaux réalistes ou de structures de petite taille sur des maillages non-uniformes entraîne inévitablement une dégradation des perfomances et donc une perte d'efficacité du schéma numérique. Dans cette thèse nous avons étudié une technique de pas de temps local combinée à une méthode GDDT-$\mathbb{P}_p$ qui permet d'adapter le pas de temps à la taille des mailles, dans l'esprit de celles proposées dans [Diaz 2009]-[Grote 2009]. Il s'agit là d'une étude préliminaire qui a fourni des résultats encourageants mais qui se doit d'être prolongée dans plusieurs directions. Tout d'abord une étude de stabilité complète doit être menée (i.e. qui considère toutes les configurations possibles de conditions aux limites, ainsi que la propagation dans un milieu conducteur). Cette étude théorique devrait être complétée par une analyse de précision a priori. Ensuite, il serait souhaitable d'étendre la technique proposée ici pour un schéma saute-mouton LF2 au cas d'un schéma LF4. Enfin, une étude numérique plus poussée est nécessaire pour mieux évaluer le comportement (précision et stabilité en temps long) et les performances computationnelles des méthodologies GDDT-$\mathbb{P}_p$ à pas de temps local résultantes.
- Concernant notre contribution sur l'exploitation d'architectures de calcul multi-GPU pour les si-

mulations numériques tridimensionnelles à base de la méthode GDDT-$\mathbb{P}_p$, en vue d'augmenter la scalabilité forte, il serait intéressant d'étudier une stratégie de distribution des données reposant sur un partitionnement hiérarchique (multi-niveau) du maillage de calcul afin de favoriser au maximum la localité des données et minimiser les échanges inter-GPU. Par ailleurs une exploitation des possibilités de CUDA 4.1 (telle que les accès lecture/écriture directs entre mémoires GPU) devrait permettre d'augmenter les performances globales. Enfin, la prise en compte d'une stratégie de pas de temps local et le passage à une arithmétique double précision nécessiteront certaines adaptations algorithmiques.

Annexe A

Annexe

A.1 Generalities about GPUs

Many-core GPUs are currently capable of delivering over 1 TFlops of single precision performance versus 100 GFlops for the multi-core CPUs and the GPU peak performance continues to improve at a rate substantially higher than CPU performance. This performance gap is mainly due to the architectural differences between the two types of processors (see Fig. A.1). CPU cores continue to be optimized for single threaded performance at the expense of parallel execution. Large CPU cache memories are provided to reduce the instruction and data access latencies of large complex applications ; but as additional thread contexts share the limited resources of cache capacity and external memory bandwidth, throughput and power efficiency decrease. As the execution bandwidth of a processor module increases due to higher core count, memory bandwidth climbs at a slower rate. In addition, the cache capacity available per thread of execution decreases as more threads are packed into the same limited silicon. As a result, when executing code consisting of many sequential operations of the same type, like scientific workloads, most of the complex circuitry on a CPU, i.e. those caches, instruction decoders, branch predictors, and other features that are not architecturally visible but which enhance single-threaded performance cause both decreased system throughput and power consumption. In contrast, the design philosophy of the GPUs is motivated by the ability to optimize for the execution throughput of massive numbers of threads. The hardware takes advantage of a large number of execution threads to find work to do when some of them are waiting for long latency memory accesses, thus minimizing the control logic required for each execution thread. Small cache memories are provided to help control the bandwidth requirements of these applications so multiple threads that access the same memory data do not need to all go to the dynamic random access memory (DRAM).

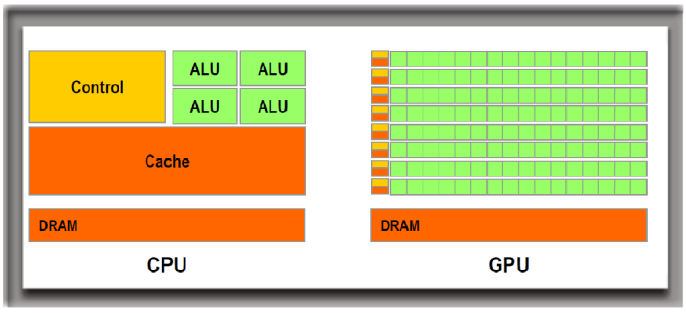


Figure A.1 – GPU vs CPU design philosophies.

A.1.1 Evolution of GPU computing with CUDA

CUDA (Compute Unified Device Architecture) [NVIDIA 2011] is a parallel computing architecture developed by NVIDIA, which enables non-graphics computations on GPUs. On a CUDA-enabled GPU, many processes can be executed at once, which enables developers to conveniently take advantage of a data parallel-programming methodology. CUDA-enabled GPUs of the same generation have in general the same architectural design.

A.1.1.1 G80

Introduced in November 2006, G80 was the first generation NVIDIA graph processor based upon a unified shader architecture. That is, all calculations are performed through scalar unified shader pipelines also known as Streaming Processors (SPs). NVIDIA calls an individual SP a single processing core. It is a fully pipelined, single-issue, in-order microprocessor complete with 2 Arithmetic Logic Units (ALUs) and a Floating Point Unit (FPU) but it does not have any register or cache. G80 consists of 128 processing cores gathered into 8 Texture Processor Clusters (TPCs), grouped in a higher level entity called a Streaming Processor Array (SPA), represented on Fig. A.2. Every such TPC is subdivided for 2 sub-clusters called Streaming Multiprocessors (SMs) shown on Fig. A.3. A SM is an array of 8 SPs executing the resident threads' instructions and supporting 32-bit precision integer and floating-point operations, along with 2 more processors called Special Functions Units (SFUs). Each SFU has 4 FP multiply units which are used for transcendental operations (e.g. sin, cosine ...) and interpolation, the latter being used in some of the calculations for things like anisotropic texture filtering. There is also a multithreaded issue unit that dispatches instructions to all of the SPs and SFUs in the group. In addition to the processor cores in a SM, there is a very small instruction cache, a read only data cache and a 16KB of low-latency shared on-chip memory, which is similar in speed to a typical L1 cache. It is, however, an explicitly managed RAM available to all threads. The shared memory was one of the G80's key innovations as it was instrumental in accelerating matrix multiplications, the basis of many mathematics and physics algorithms. A constant cache memory of the first level offers additional memory and can accelerate accesses to read-only constant data (64KB in total, *i.e.* 8KB per cluster). Finally, threads in a SM can access texture units composed of 4 texture address units and 8 texture filtering units (see Fig. A.3) which can perform various image filtering operations. A read-only texture cache is shared by all the processors and speeds up reads from the texture memory space, which is implemented as a read-only region. Texture units' caches can be used to aggregate repeated accesses to read-only arrays (128KB in total, *i.e.* 16 KB per cluster). The global, constant, texture and local memory are optimized for different memory usages and reside on the off-chip device memory, accessible by all SPs. Each multiprocessor can support up to 768 co-resident threads having all access to a register unit with a 32KB register file. The hardware handles all thread management and scheduling, with effectively no overhead. The GPU connects to other components via the PCI-Express (PCIe) bus, which provides the channel for transferring data between the system memory and the GPU's on-board memory.

A.1.1.2 GT200

In June 2008, NVIDIA introduced a major revision to the G80 architecture : the second generation Graphics Tesla architecture GT200. NVIDIA's GT200 GPU has a significant increase in computational power thanks to its 240 SPs, up from 128 in the previous G80 design. In G80 the SPA was made up of 8 TPCs, but with GT200 we have moved up to 10 (see Fig. A.4). NVIDIA purposefully designed its GPU architecture to be modular, so a single TPC can be made up of any number of SMs.

In the G80 architecture it was made up of 2 SMs but with the GT200 architecture it now has 3 SMs (see Fig. A.5). The components of the TPC have not changed, a TPC is made up of SMs, some control logic and a texture block (see Fig. A.6). With the move to GT200, NVIDIA doubled the number of texture address units for achieving a one to one ratio of address/filtering units, improved significantly the texture filtering units performance and increased the ratio of SPs to texture processors. Indeed, a SM is a total of 8 SPs and 2 SFUs, so that brings the total up to 24 SPs and 6 SFUs per cluster in GT200 (up from 16 SPs and 4 SFUs in G80). As GT200 series GPUs are designed with 3 SMs per cluster (every cluster in G80 contains 2 SMs), the L1 cache has got a size increase from 16KB to 24KB per cluster (240KB in total). For a 87,5% increase in compute, there is a mere 25% increase in texture processing power.

One of the major new features in GT200 is the ability to process double precision floating point data in hardware. The GT200 features a single dedicated double precision (DP) FPU for each core, versus 8 single precision FPUs. The size of the register file for each SP array has been doubled (64KB, up from

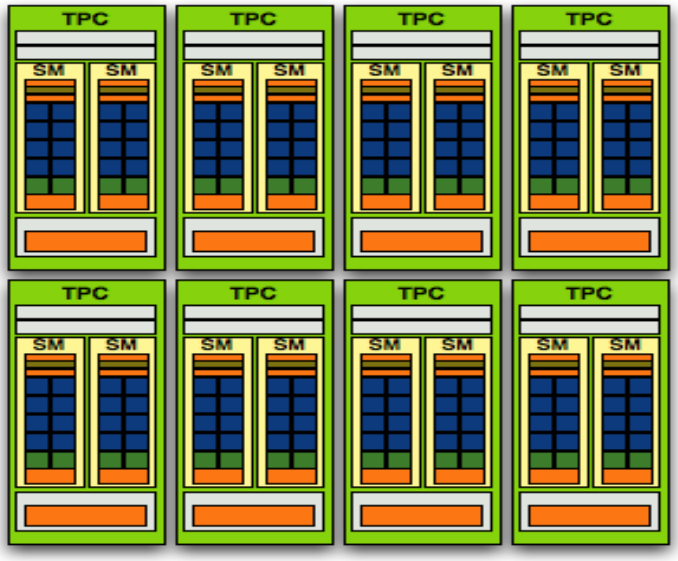


Figure A.2 – NVIDIA's G80 Streaming Processor array (SPA) : 128 SPs in 8 TPCs.

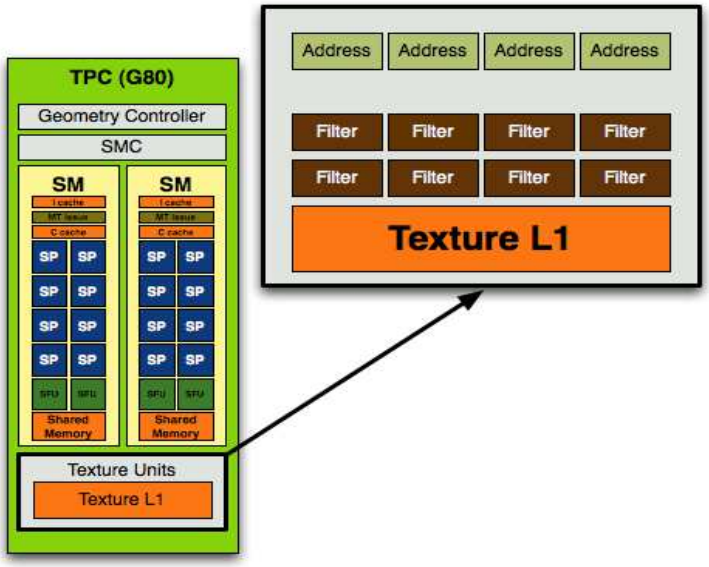


Figure A.3 – NVIDIA's G80 Texture Processor Cluster (TPC).

32KB in the previous G80 design), allowing a greater number of threads to execute on-chip at any given time. Each thread gets its own dedicated set of 32-bit registers, which lets the hardware freely schedule runnable threads without needing to save/restore any thread state.

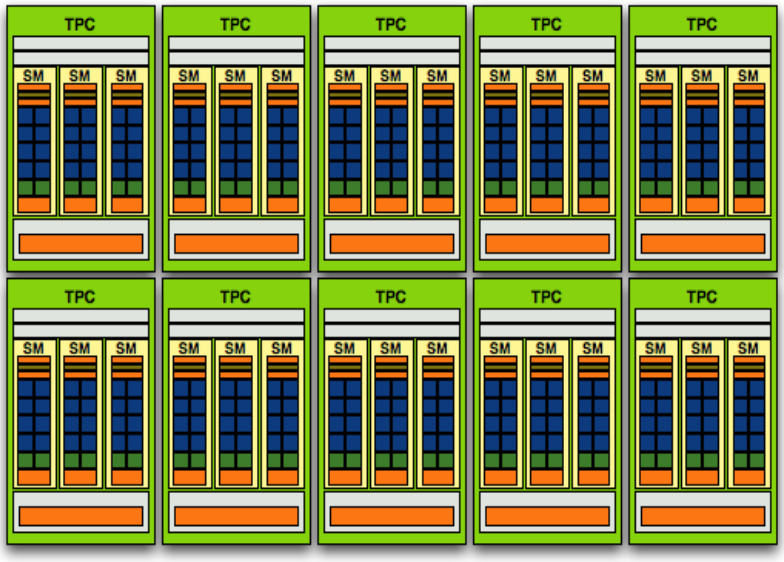


Figure A.4 – NVIDIA's GT200 Streaming Processor array (SPA) : 240 SPs in 10 TPCs.

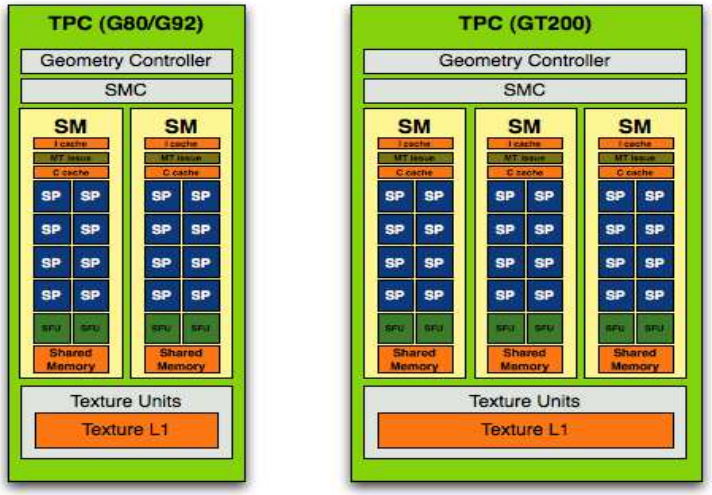


Figure A.5 – G80 and GT200 TPCs.

A high-end GT200 GPU, such as the Tesla C1060 (see Fig. A.7), features 240 SPs running at 1.3 GHz. It has got a 512-bit memory interface to 4GB (8 memory channels of 64-bit each) of GDDR3 running at 102.4 GB/s and processes 933 GFlops and 77 GFlops in single and double precision floating point arithmetic respectively. However dramatic increases in computing performance can only be achieved after extensive optimization and tuning, by rewriting processing intensive parts with parallelization techniques. Whereas CPUs are primarily designed to maximize the rate at which many thousands of parallel threads

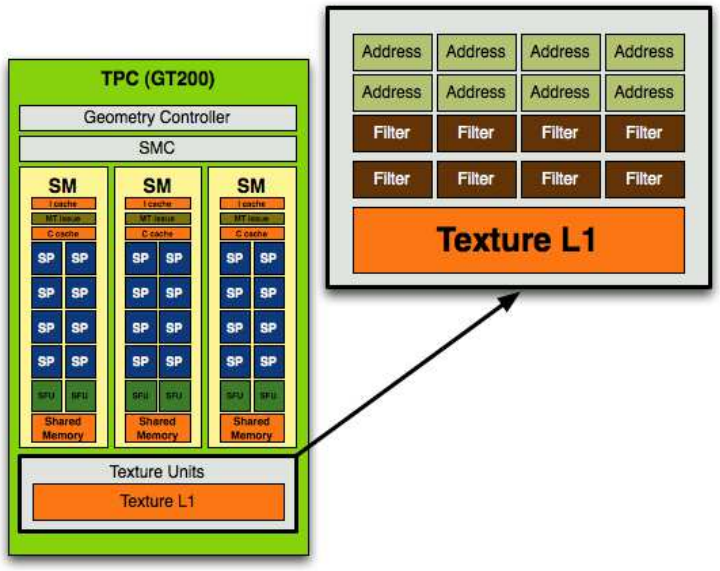


Figure A.6 – NVIDIA's GT200 Texture Processor Cluster (TPC).

can complete their work. And, unlike more traditional microprocessors, GPUs rely on multithreading, as opposed to a cache, to hide the latency of transactions with external memory. It is therefore necessary to design algorithms that create enough parallel work to keep the machine fully utilized.

We recommend two good reference books about starting developing CUDA C for General-Purpose GPU Programming, « Programming Massively Parallel Processors : A Hands-On Approach »([Kirk 2010]) by David Kirk and Wen-Mei Hwu which teaches general parallel principles to developing parallel applications that perform well on NVIDIA's GPUs using NVIDIA's CUDA language ; and « CUDA by Example : An Introduction to General-Purpose GPU Programming »([Sanders 2010]), by Jason Sanders and Edward Kandrot's, which provides a quick-start guide and an overview of techniques for interfacing with GPU hardware using CUDA C.

A.1.1.3 Fermi

While G80 was a pioneering architecture in GPU computing and GT200 a major refinement, their designs where nevertheless deeply rooted in the world of graphics. With its combination of ground breaking performance functionality and programmability, the Fermi architecture introduced in March 2010 represents the next revolution in GPU computing. At a high level, Fermi does not look like much different than a bigger GT200. Despite the similarities, large parts of the architecture has evolved. The redesign happened at the core lower level. NVIDIA used to call these SPs Streaming Processors, now they call them CUDA cores. A Fermi-based GPU features up to 512 CUDA cores organized in 16 SMs of 32 CUDA cores each, that is a fourfold increase over GT200. Every SM is accommodated with 2 warp schedulers, 2 dispatch units, 4 SFUs, 16 load/store units, a register unit with a large register file (see Fig. A.8).

The already massive storage resources for each core have been doubled to support the second pipeline, Fermi's register file increased to 128KB total. To utilize those execution resources, the number of threads in flight for each Fermi core has increased to 1536 (up from 1024 in GT200 and 768 in G80). In GT200 and G80, half of a warp was issued to an SM every clock size. In other words, it takes two clocks to issue a full 32 threads to a single SM. With the Fermi generation, execution within an SM occurs at the granularity of a warp, which is a set of 32 threads, and each SM features 2 warp schedulers and 2

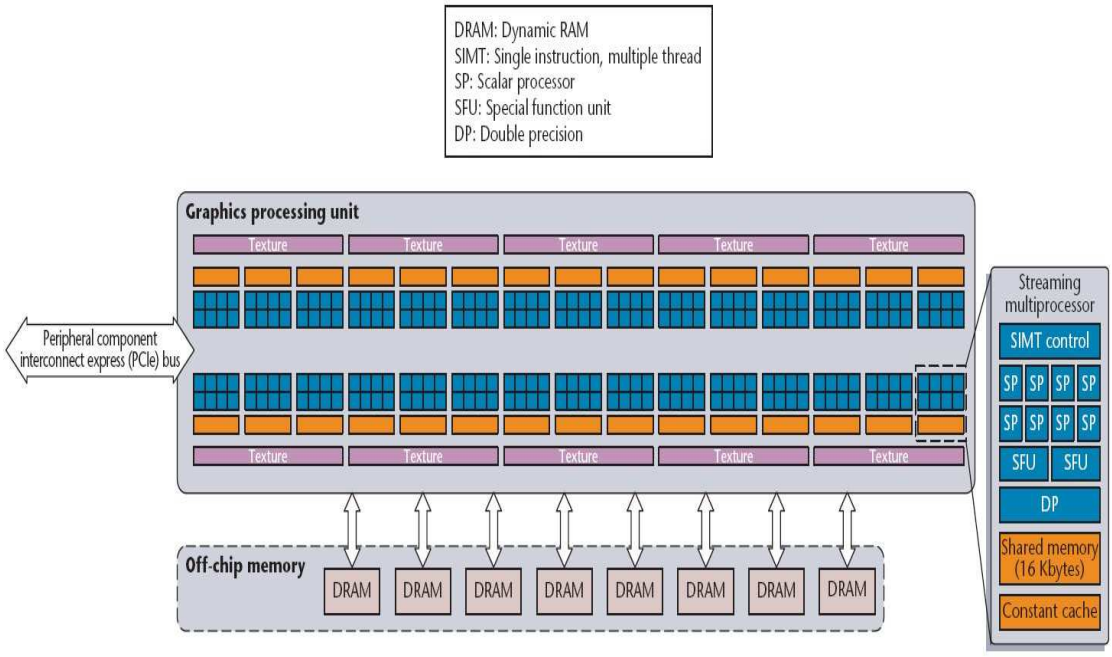


Figure A.7 – Block diagram of a high-end GT200 GPU. This Tesla C1060 contains 240 cores organized in 30 multiprocessors.

instruction dispatch units (see Fig. A.9), allowing 2 warps to be issued and executed concurrently. The dual warp scheduler selects 2 warps, and issue one instruction from each warp to a group of 16 cores, 16 load/store units or 4 SFUs. In Fermi, each core can have up to 48 warps in-flight at once (up from 32 in GT200 and 24 in G80). Because warps execute independently, Fermi's scheduler does not need to check for dependencies from within the instruction stream.

Another aspect of improved performance and accuracy are the streaming multiprocessors (SMs), which have four times as many CUDA cores as the predecessor (see Fig. A.8). Each CUDA processor has a fully pipelined arithmetic logic unit (ALU) and floating-point unit (FPU). Prior GPUs used IEEE 754-1985 floating-point math, but Fermi implements the IEEE 754-2008 standard. Thus it provides the fused multiply-add (FMA) instruction for both single- and double-precision maths. FMA improves over a multiply-add (MAD) instruction by doing the multiplication and addition with a single final rounding step with no loss of precision in the addition. Note also that Fermi doubles the number of SFUs (see Fig. A.8), which execute transcendental instructions such as sine, cosine, reciprocal and square roots.

As shown in Fig. A.10, Fermi is a big change from GT200, especially for the memory hierarchy. The cores in GT200 are not complete and every group of three cores shares a TPC. Fermi does away with this arrangement and gives each core its own load store unit (LSU) and its (semi-coherent) L1 data cache. NVIDIA organizes these SMs into 4 Graphics Processing Clusters (GPCs) represented on Fig. A.11. A GPC is basically a macro-block that subsumes at most 4 SMs. Surrounding the 4 GPCs are six 64-bit memory controllers, for a 384-bit memory interface (like in G80), supporting up to 6GB of GDDR5 DRAM memory to facilitate high bandwidth access to the frame buffer (see Fig. A.10).

The external interface to the rest of the system is unchanged, still relying on PCI-Express 2.0, but the controller logic has improved (see Fig. A.12). Fermi can now simultaneously transfer data to and from the host, which enables computation on the GPU to be better pipelined with respect to the rest of the system.

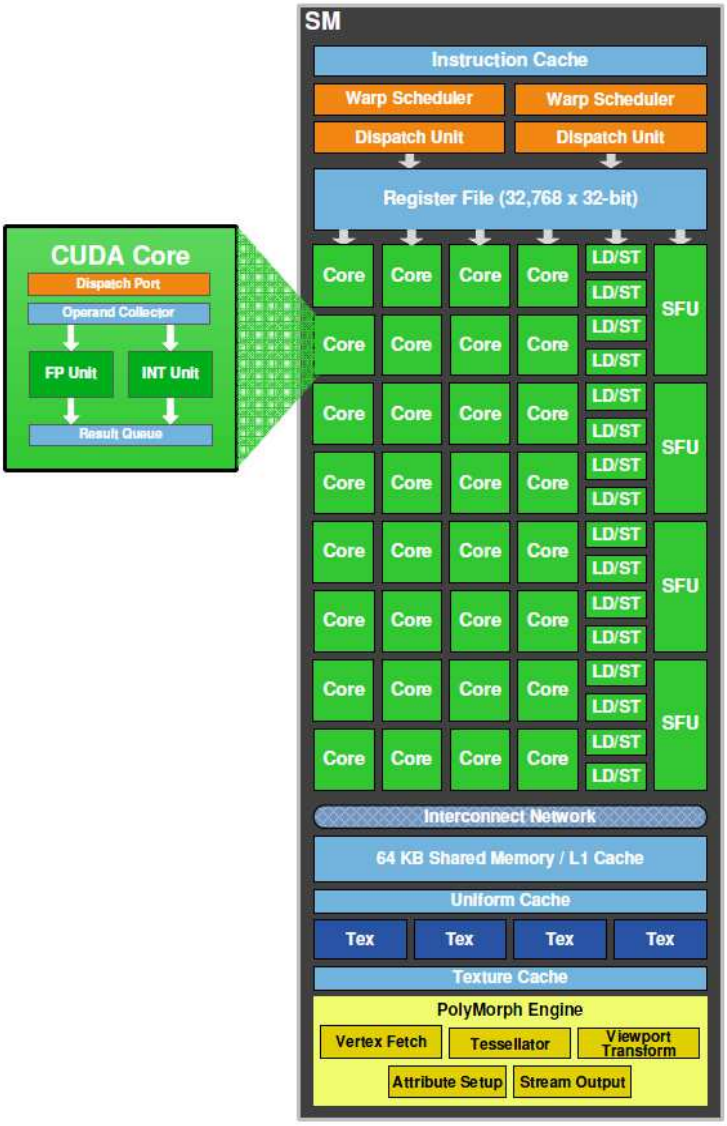


Figure A.8 – Streaming Multiprocessor (SM) of a Fermi-based GPU.

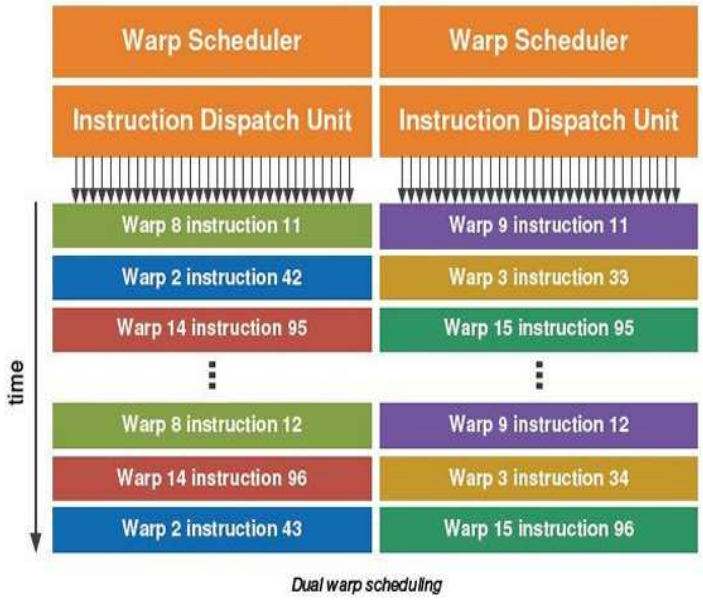


Figure A.9 – Double wrap scheduling.

The control hierarchy is similar to the GT200, with a global scheduler that issues work to each SM. Previously the global scheduler (and hence the GPU) could only have a single kernel in flight. NVIDIA's newer scheduler can maintain state for up to 16 different kernels, one per SM. Each SM runs a single kernel, but the ability to keep multiple kernels in flight increases utilization especially when one kernel begins to finish and has fewer block left. More importantly, assigning a kernel per core means that smaller kernels can be efficiently dispatched to the GPU.

Finally, the Gigathread global scheduler (see Fig. A.11) distributes thread blocks to the SM thread schedulers (see Fig. A.13). Attached to each SM is 64KB of configurable partitioning of shared memory and a semi-coherent L1 cache (see Fig. A.14), giving each core a real memory hierarchy. It can be partitioned as 16KB/48KB or 48KB/16KB ; one partition means that applications written for GT200 that require 16KB of shared memory will still work just fine on Fermi. GT200 did have a L1 texture cache (one per TPC), but the cache was mostly useless when the GPU ran in compute mode. The L1 cache on Fermi works as a counterpart to shared memory : while shared memory improves memory access for algorithms with well defined memory access, the 48KB L1 cache configuration offers greatly improved performance over direct access to DRAMs for programs whose memory accesses are not known beforehand. When configured with 48KB of shared memory, programs that make extensive use of shared memory can perform up to three times faster, especially for problems that are bandwidth constrained. Tying together all the GPCs is an 768KB shared L2 cache (see Fig. A.14), which is the memory agent and can assist with some synchronization. Unlike the 256KB cache on GT200 series GPUs, which is used for fast storage of texture information, the L2 cache on Fermi is a fully read/write, coherent write-through cache for all data formats. As a consequence, with up to 48KB of L1 cache per SM and a global L2 cache, threads can access the same memory locations at runtime automatically faster, irrespective of the choice of algorithm.

In the Fermi architecture, as many as 16 double-precision fused multiply-add (FMA) operations can be performed per SM per clock. In practical terms, this means that a typical double-precision matrix runs approximately four times faster than in the GT200 implementation. Put another way, double-precision previously took eight times as long as single-precision calculations, but now it is just half of the time. Due to this speed-up, highly double-precision bound application will enjoy a huge benefit, up to eight times the peak performance over GT200.

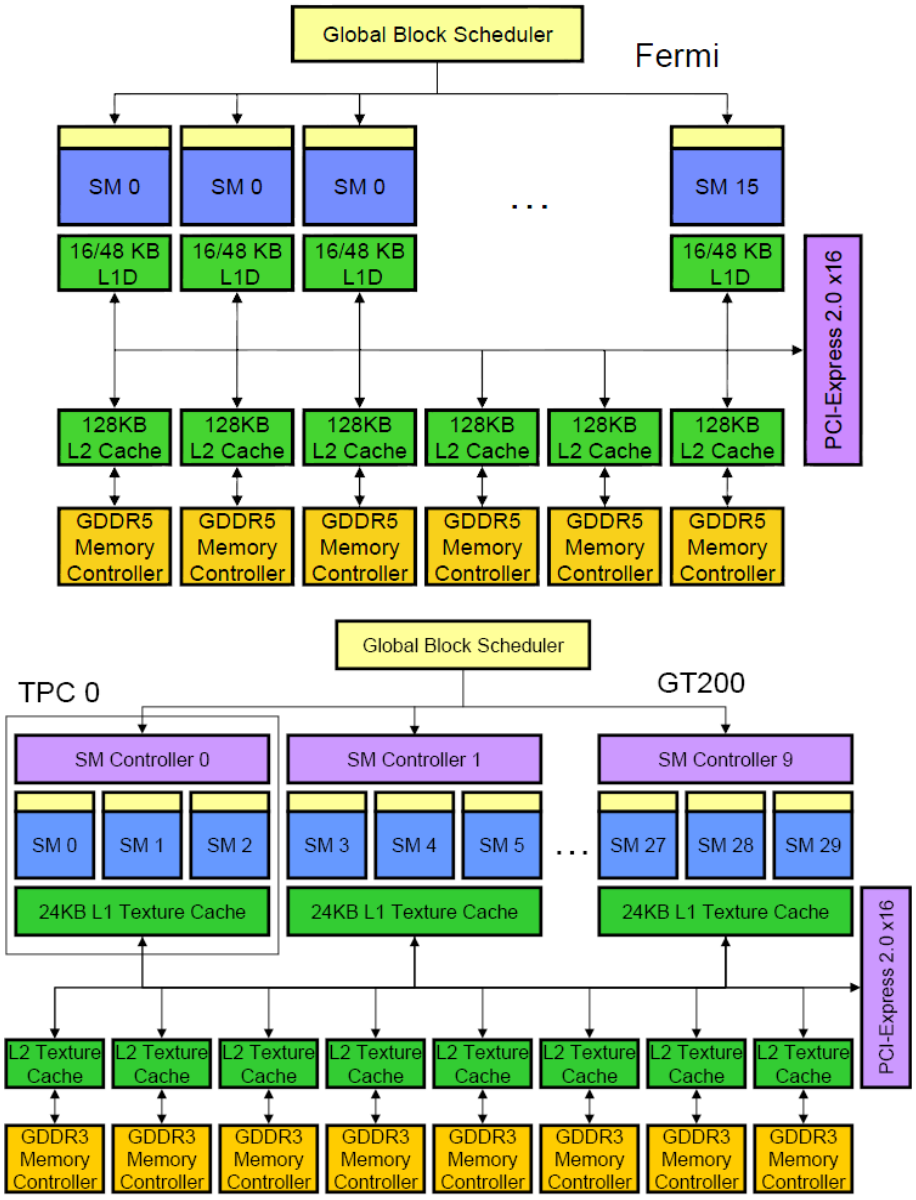


Figure A.10 – Fermi and GT200 overview.

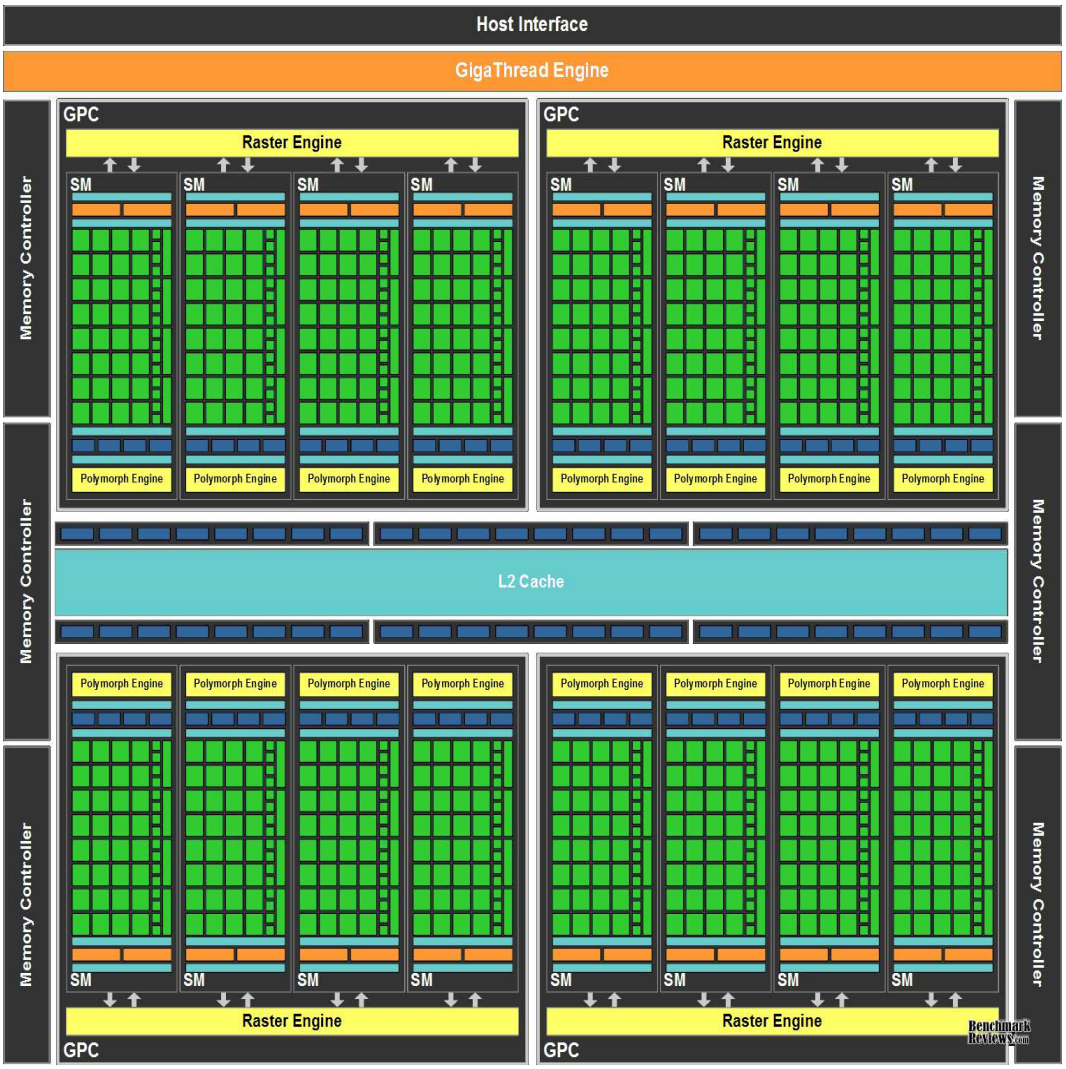


Figure A.11 – Fermi block diagram. The GPU is designed to be highly modular, enabling a variety of different products to be built.

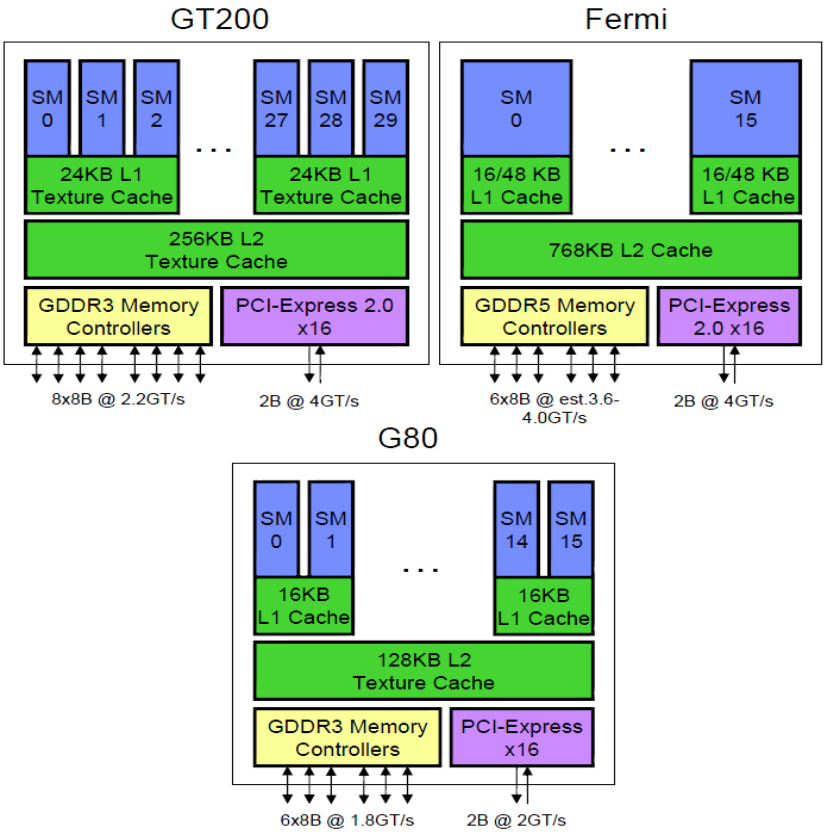


Figure A.12 – NVIDIA's G80, GT200 and Fermi system architectures.

In the GT200 architecture, up to 3 SMs shared one texture engine containing 8 texture filtering units grouped together in TPCs. On a Fermi-based GPU, each SM has 4 dedicated texture units and a dedicated texture cache, eliminating the need for TPCs. Also the internal architecture of the texture units has been significantly enhanced. The goal with Fermi was to improve delivered texture performance through improved efficiency. This was achieved by moving the texture units within the SM, improving the efficiency of the texture cache, and higher clock speed. Fermi can't actually support as many threads in parallel as GT200. NVIDIA found that the majority of computes cases were bound shared memory size, not thread count in GT200. Thus thread count went down, and shared memory size went up in Fermi.

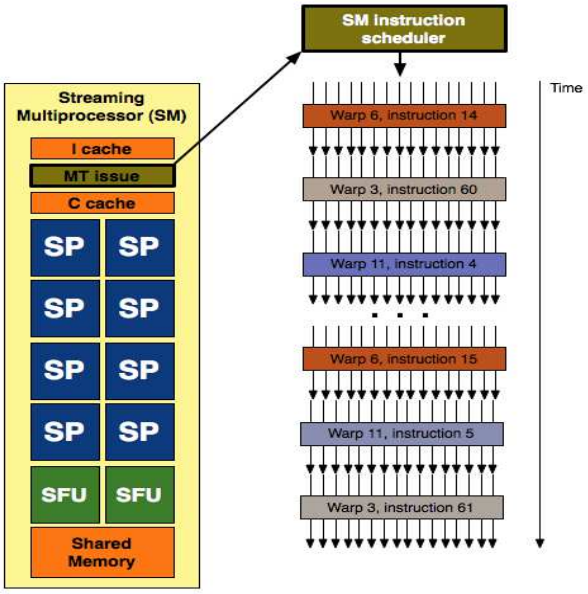


Figure A.13 – Streaming Multiprocessor architecture and scheduler.

An overview of the different features for G80, GT200 and Fermi gathered so far is shown below in Tab. A.1.1 and A.1.2.

GPU Chip Series	TPCs	SMs per TPC	Cores per SM			Warps schedulers per SM	Warps in flight per SM	Max threads per SM	Max threads per block	Max threads in flight
			SP	DP	SFU					
G80	8	2	8	0	2	1	24	768	512	12288
GT200	10	3	8	1	2	1	32	1024	512	30720
Fermi	16	1	32	16	4	2	48	1536	1024	24576
			16 Load/Store							

Table A.1.1 – Overview of G80, GT200 and Fermi features

GPU	G80	GT200	Fermi
Transistors	681 million	1.4 billion	3.0 billion
CUDA Cores	128	240	512
Double-Precision FP Capability	None	30 FMA ops/clock	256 FMA ops/clock
Single-Precision FP Capability	128 MAD ops/clock	240 MAD ops/clock	512 FMA ops/clock
Shared Memory (per SM)	16 KB	16 KB	Config. 48 KB or 16 KB
Shared Memory banks (per SM)	16	16	32
L1 cache (per SM)	16 KB	24 KB	Config. 48 KB or 16 KB
L2 cache	192 KB	256 KB	768 KB
Register file size	32 KB	64 KB	128 KB
Concurrent kernels	No	No	Up to 16
Load/Store Address Width	32-bit	32-bit	64-bit
Size of global memory transactions	32/64/128 B	32/64/128 B	32/128 B

TABLE A.1.2 – Overview of G80, GT200 and Fermi features

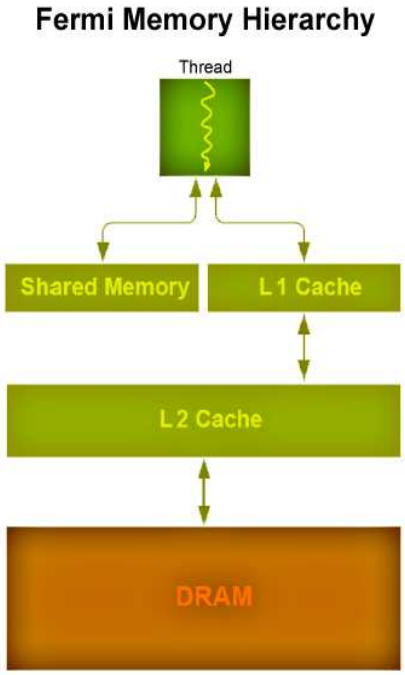


Figure A.14 – Fermi memory hierarchy.

A.1.2 Parallel Execution

To a CUDA programmer, the computing system consists of a host, which is a traditional central processing unit (CPU), and one or more devices, the GPUs, viewed as massively parallel processors equipped with a large number of arithmetic execution units. Kernels are a blocked *Single Program, Multiple Data* (SPMD) parallel computation model ; a kernel is a data parallel function that executes a single sequential program across many parallel GPU threads in a *Single Instruction, Multiple Data* (SIMD) fashion.

Launching a kernel for GPU execution is similar to calling the kernel function, except that the programmer needs to specify the space of GPU threads that execute it, called a grid. A grid contains multiple thread blocks having the same number of threads, organized in a two-dimensional space, executing the same kernel. Each thread block is, in turn, organized as a three-dimensional array of threads with a total size up to 512 threads for GT200 and 1024 threads for Fermi (see Tab. A.1.1). Each GPU thread is assigned a thread block index accessed within the kernel through the built-in blockIdx variable, and a thread index accessed through threadIdx. A unique thread ID is assigned to each thread, computed from both thread indexes, allowing all threads in a grid to distinguish themselves from each other and to identify the appropriate portion of the data to process. A single thread block is guaranteed to be resident on a single multiprocessor so threads in the same thread block may coordinate their activities by using shared memory, and by synchronizing at a barrier using a specific intrinsic function. This allows the threads that execute the function call to be held at the calling location until every thread in the block reaches the location. The ability to synchronize also imposes execution constraints on threads within a block. These threads should execute in close time proximity with each other to avoid excessively long waiting times. CUDA runtime systems satisfy this constraint by assigning execution resources to all threads in a block as a unit ; that is, when a thread of a block is assigned to an execution resource, all other threads in the same block are also assigned to the same resource. This ensures the time proximity of all threads in a block and prevents excessive waiting time during barrier synchronization. By not allowing threads in different blocks to perform barrier synchronization with each other, the CUDA runtime system can execute blocks in any order relative to each other because none of them must wait for each other.

When a CUDA application is executed, the CUDA runtime system generates the corresponding grid of threads which are assigned to execution resources on a block-by-block basis. When a multiprocessor is given one or more thread blocks to execute, it creates, manages, schedules, and executes threads in groups of 32 parallel threads called warps (see Fig. A.13). The way a thread block is split into warps is static, each warp consists of 32 threads of consecutive index values : the first 32 threads form the first warp, thread 32 through 63 the second warp, so on and so forth. When an instruction executed by the threads in a warp must wait for the result of a previously initiated long-latency operation, the warp is not selected for execution. Another resident warp that is no longer waiting for results is selected for execution. If more than one warp is ready for execution, a priory mechanism is used to select one for execution, This mechanism of filling the latency of expensive operations with work from other threads is often referred to as latency hiding. Note that warp scheduling is also used for tolerating other types of long-latency operations such as pipelined floating-point arithmetic and branch instructions. With enough warps around, the hardware will likely find a warp to execute at any point in time, thus making full use of the execution hardware in spite of these long-latency operations. The selection of ready warps for execution does not introduce any idle time into the execution time line, which is referred to as zero-overhead thread scheduling. With warp scheduling, the long waiting time of warp instructions is hidden by execution instructions from other warps. At any time, the SM executes only a subset of its resident warps for execution. This allows the other warps to wait for long-latency operations without slowing down the overall execution throughput of the massive number of execution units.

So the SM executes a warp in a SIMD fashion, by scheduling each thread in the warp to a SP. Each SM can perform computation independently but the SP cores within single multiprocessor all execute instructions synchronously. NVIDIA call this paradigm SIMT or *Single Instruction, Multiple Thread*. This SIMT architecture is similar to SIMD (*Single Instruction, Multiple Data*), SIMT instructions specify the execution and branching behaviour of a single thread. Unlike SIMD, SIMT offers substantial efficiencies by enabling programmers to write thread-level parallel code for interdependent scalar threads and data-parallel code for coordinated threads. SIMT thread execution is largely invisible to the CUDA programmer because the processor transparently handles the case in which different threads of a warp want to follow divergent execution paths. However, much like cache line sizes on traditional CPUs, programmers can ignore it when designing for correctness but must consider it carefully when designing for peak performance.

CUDA has hierarchical cal organization into sequential threads, with multiple threads per block and multiple blocks per kernel, and defines the hierarchy of scopes at which threads can share memory and synchronize. A CUDA program is a unified source code encompassing both host and device code. The ability to execute the same application code on hardware with different numbers of execution resources is referred to as transparent scalability ; for CUDA programs, this is allowed by the lack of synchronization constraints between blocks. Once the grid of GPU threads that executes the kernel is specified, the programmer needs to allocate memory on the device and transfer pertinent data from the host memory to the allocated device memory. Similarly, after device execution, the programmer needs to transfer result data from the device memory back to the host memory and free up the device memory that is no longer needed. CUDA programming model generally ends with memory optimizations in the kernel, such as utilizing the shared memory and coalescing accesses to the global memory (coalescing is a result of global memory access of 32, 64, or 128- bit words by half wraps of threads for GT200 and of 32 or 128-bit words by warps of threads for Fermi, see Tab. A.1.2), as well as other optimizations in the kernel in order to achieve an optimal balance between single-thread performance and the level of parallelism.

Simple CUDA kernels generally achieve only a small fraction of the potential speed of the underlying hardware due to the fact that global memory, which is typically implemented with Dynamic Random Access Memory (DRAM) of a GPU (see Fig. A.7), tends to have long access latencies (hundred of clock cycles) and finite access bandwidth. Although having many threads available for execution can theoretically tolerate long memory access latencies, one can easily run into a situation where traffic congestion in the global memory access paths prevents all but a few threads from making progress, thus

rendering some of the streaming multiprocessors idle. In order to circumvent such congestion, CUDA provides a number of additional methods for accessing memory that can remove the majority of data requests to the global memory.

The compute to global memory access (CGMA) ratio is defined as the number of floating-point calculations performed for each access to the global memory within a region of a CUDA program. The CGMA ration has major implications on the performance of a CUDA kernel. For example, the NVIDIA GT200 supports 150 gigabytes per second (GB/s) of global memory access bandwidth (177 GB/s for Fermi). The highest achievable floating-point calculation throughput is limited by the rate at which the input data can be loaded from the global memory. With 4 bytes in each single-precision floating-point datum, one can expect to load not more than 37.5 (150/4) giga single-precision data per second. With a CGMA ration of 1.0, the kernel will execute at no more than 37.5 billion floating-point operations per second (gigaflops), as each floating-point operation requires one single-precision global memory datum. Although 37.5 GFlops is a respectable number, it is only a tiny fraction of the peak performance of 1000 GFlops for the GT200. Fortunately, as we previously noted, CUDA supports several types of memory that can be used by programmers to achieve high CGMA ratios and thus high execution speeds in their kernels.

A.1.3 CUDA device memory model

Fig. A.15 shows these CUDA device memories. The constant memory supports short latency, high bandwidth and read only access by the device. The texture memory supports read only access by the device and offers different addressing modes, as well as data filtering, for some specific data formats. The constant and texture memory are also accessible from the host.

Registers and shared memory are on-chip memories. Variables that reside in these types of memory can be accessed at very high speed in a highly parallel manner. Registers are allocated to individual threads ; each thread can read and/or write its own registers very fast with almost no delay. A kernel function typically uses registers to hold frequently accessed variables that are private to each thread. Shared memory is allocated to thread blocks ; all threads in a block can access variables in the shared memory locations allocated to the block but the shared memory cannot be shared between other blocks. Data in shared memory has a lifetime of the thread block. Shared memory acts like a low-latency, high-bandwidth software managed cache memory and is almost as fast as the registers. It is an efficient means for threads to cooperate by sharing their input data and the intermediate results of their work.

The largest memory location is the global memory. Globally, all threads have read and write access to the global memory. Local variables in a kernel function are automatically allocated in registers (or local memory). Variables in other GPU memories must be created and managed explicitly, through the CUDA runtime Application Programming Interface (API). By declaring a CUDA variable in one of the CUDA memory types, a CUDA programmer dictates the visibility and access speed of the variable.

As illustrated in Fig. A.16, in the CUDA parallel mode, each thread has a per-thread private memory space used for register spills, function calls, and C automatic array variables. Each thread block has a per-block shared memory space used for inter-thread communication, data sharing, and result sharing in parallel algorithms. Grids of thread blocks share results in global memory space after kernel-wide global synchronization.

The GPU architecture poses various constraints on the number of threads and thread blocks that can be simultaneously scheduled on an SM. First, as shown in Tab. A.1.1, the maximum number of threads, thread blocks and warps on an SM is for instance 1024, 8 and 32 respectively in the NVIDIA GT200 design and 1536, 8 and 48 in the NVIDIA Fermi design. Second, a thread block can contain at most 512 threads in GT200 and 1024 in Fermi (see Tab. A.1.1). Last registers and shared memory for each SM are dynamically partitioned among the batch of thread blocks scheduled on the SM, so their sizes poses a limit on the number of threads and thread blocks in a batch. While the block arrangement does not have

a significant impact on the execution of threads on multiprocessors, it strongly affects the manner in which threads access memory and use the memory optimisations available on the GPU. These hardware constraints are the main factor that makes the optimization of a CUDA program time-consuming because they result in a discontinuous optimization space.

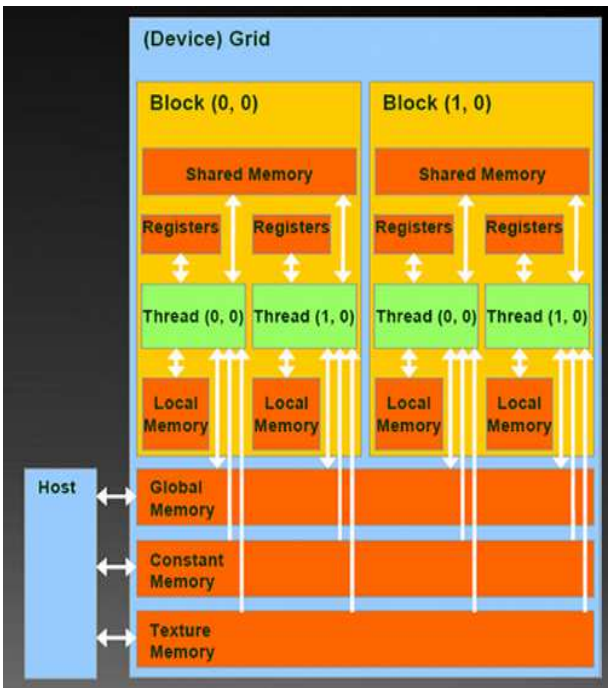


Figure A.15 – Overview of the CUDA device memory model.

Variable Declaration	Memory	Scope	Lifetime
Automatic variables other than array	Register	Thread	Kernel
Automatic array variables	Local	Thread	Kernel
__device__, __shared__, int SharedVar	Shared	Block	Kernel
__device__, int GlobalVar	Global	Grid	Application
__device__, __constant__, int ConstVar	Constant	Grid	Application

Table A.1.3 – CUDA Variable Type Qualifiers

Tab. A.1.3 presents the CUDA syntax for declaring program variables into the various types of device memory. Each such declaration also gives its declared CUDA variable a scope and lifetime. Scope identifies the range of threads that can access the variable : by a single thread only, by all threads of a block, or by all threads of all grids. Lifetime specifies the portion of the program's execution duration when the variable is available for use : either within a kernel's invocation or throughout the entire application. As shown in Tab. A.1.3, all automatic scalar variables declared in kernel and device functions are placed into registers. We refer to variables that are not arrays as scalar variables. The scopes of these automatic variables are within individual threads. When a kernel function declares an automatic variable, a private copy of that variable is generated for every thread that executes the kernel function. When a thread terminates, all of its automatic variables also cease to exist. Note that accessing these variables is extremely fast and parallel, but one must be careful not to exceed the limited capacity of the register storage in the hardware implementations. Automatic array variables are not stored in registers. Instead they are stored into the global memory and incur long access delays and potential access congestion. The scope of these arrays is,

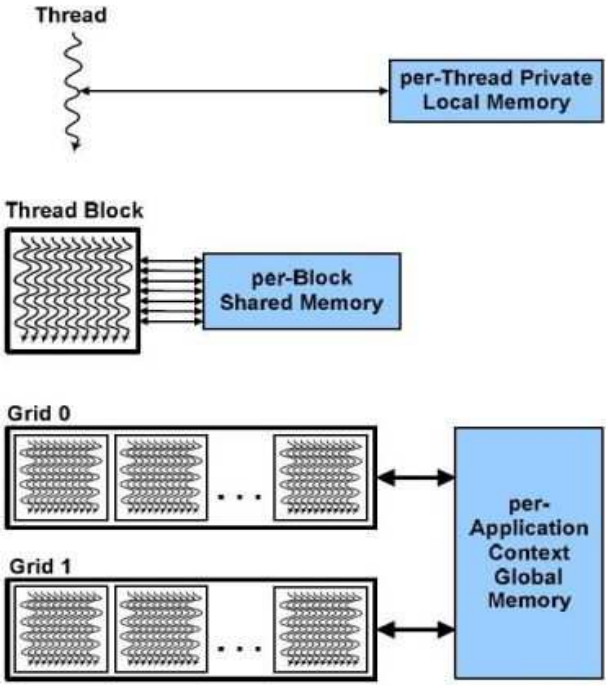


Figure A.16 – Overview of the CUDA parallel model.

like automatic scalar variables, limited to individual threads. That is, a private version of each automatic array is created for and used by every thread. Once a thread terminates its execution, the contents of its automatic array variables also cease to exist.

If a variable declaration is preceded by the keyword __shared__, it declares a shared variable in CUDA. Such declarations typically reside within a kernel function or a device function. The scope of a shared variable is within a thread block ; that is, all threads in a block see the same version of a shared variable. A private version of the shared variable is created for and used by each thread block during kernel execution. The lifetime of a shared variable is within the duration of the kernel. When a kernel terminates its execution, the content of its shared variables cease to exist. CUDA programmers often use shared memory to hold the portion of global memory data that are heavily used in an execution phase of the kernel.

If a variable declaration is preceded by the keyword __constant__, it declares a constant variable in CUDA. Declaration of constant variables must be outside any function body. The scope of a constant variable is all grids, meaning that all threads in all grids see the same version of a constant variable. The lifetime of a constant variable is the entire application execution. Constant variables are often used for variables that provide input values to kernel functions. Constant variables are stored in the global memory but are cached for efficient access. With appropriate access patterns, accessing constant memory is extremely fast and parallel.

A variable whose declaration is preceded only by the keyword __device__ is a global variable and will be placed in global memory. Accesses to a global variable are slow ; however, global variables are visible to all threads of all kernels. Their contents also persist through the entire execution. thus global variables can be used as a means for threads to collaborate across blocks. One must, however, be aware of the fact that there is currently no way to synchronize between threads from different thread blocks or to ensure data consistency across threads when accessing global memory other than terminating the current kernel execution. Therefore, global variables are often used to pass information from one kernel invocation to another kernel invocation.

Usually, most of the major CUDA releases accompanied a new GPU architecture : CUDA 1.0 was released to go with G80/G9X, CUDA 2.0 was released for GT200 in 2008 and CUDA 3.0 was released for Fermi in 2010. CUDA 4.0, released in 2011, is the first independent software-only major CUDA release, designed to make parallel programming easier with neat new features including the capability of allowing multiple GPUs to communicate directly over the PCIe bus and the creation of a unified virtual address space, named Unified Virtual Addressing (see Fig. A.17), which takes the memory space of the system and the memory spaces of the multiple GPUs in the machine and maps them as a single merged-memory space. Prior to this version, each GPU and the CPU used their own virtual address space requiring data structures to be copied to system RAM first before any additional GPUs could access the data. This means that a great deal of data has to be moved across the PCIe bus and uses memory on the host system, which may incur a performance hit due to system bandwidth and latency. Besides, separate load instructions were used to access each address space, and these have been supplanted by a single load instruction. At compile time, the addresses for these memories are determined, and the hardware is configured to correctly route fetches to the appropriate sub-space. With an unified address space, indirection for data structures is possible. NVIDIA now supports pointers and object references, which are necessary for C++ and most other high-level languages which pass by reference.

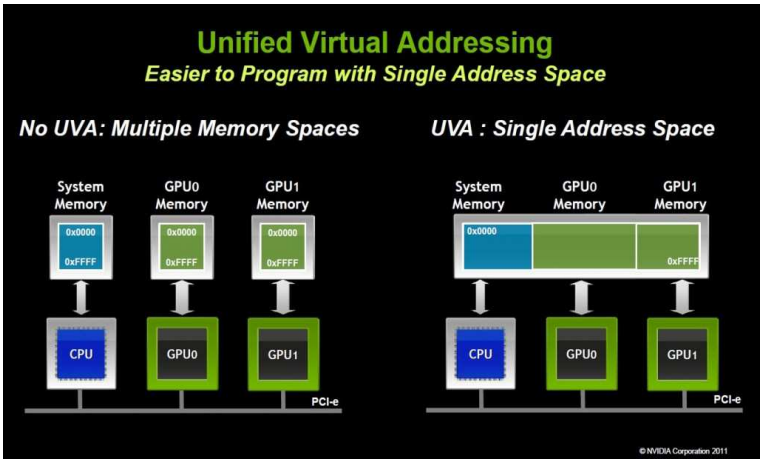


Figure A.17 – Unified Virtual Addressing

Another key new feature embedded in the CUDA 4.0 toolkit is the NVIDIA GPUDirect 2.0 technology allowing GPUs on different servers within a cluster of machines to directly access each others memory over InfiniBand links without requiring the involvement of any CPUs and a system memory copy first (see Fig. A.18). This peer-to-peer communication amongst GPUs (over the PCIe bus within a system as well as between GPUs linked to each other over InfiniBand links that lash together multiple servers) eliminates the need to manually maintain memory buffer on the host for inter-GPU exchanges and cuts the data movement across the PCIe bus in half (see Fig. A.19), improving both ease and performance of multi-GPU programming.

Today there are two GPUDirect versions - version 1.0 and version 2.0. Version 1.0 is for accelerating GPU communications between GPUs located on separate servers over InfiniBand and exists in CUDA 3.0 and in CUDA 4.0. GPUDirect version 2.0 is for accelerating GPU communications between GPUs on the same server and on the same CPU chipset and exists in CUDA 4.0.

Prior to CUDA 2.2, CUDA kernels could not access host system memory directly. For that reason, CUDA programmers used the following design pattern :

1. Move data to the GPU,
2. Perform calculation on GPU,

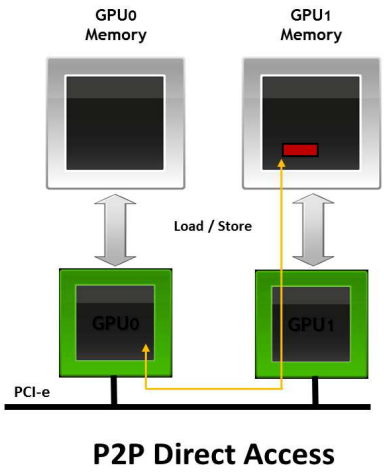


Figure A.18 – Peer-to-Peer Direct Access between GPUs enabled by NVIDIA GPUDirect v2.0.

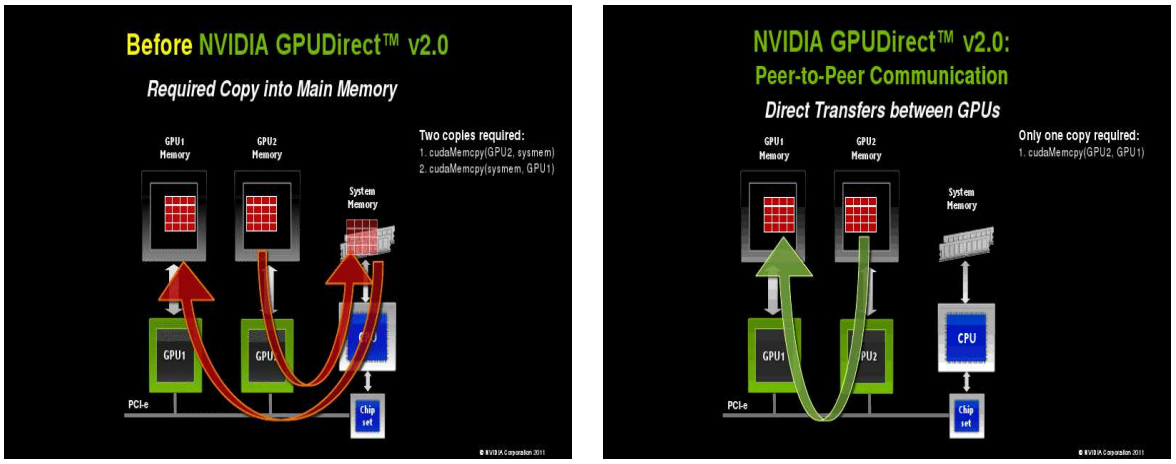


Figure A.19 – Data transfer between GPUs before and after NVIDIA GPUDirect v2.0.

3. Move result(s) from the GPU to host.

Some features in CUDA 2.2 changed this data movement paradigm by allowing APIs for mapped, transparent data transfers between the host and GPU(s). The introduction of « mapped »pinned system memory allows compute kernels to share host memory and provides zero-copy support for direct access to host system memory when running on many CUDA-enabled graphic processors. Besides new memory management functions (cuMemHostAlloc and cudaHostAlloc) enable pinned memory to be « portable »(available to all GPUs), « mapped »(mapped into the CUDA address space), and/or « write combined »(not cached and faster for the GPU to access). This allow the CUDA programmer to make data sharing between the host and graphic processor(s) more efficient by exploiting asynchronous operation, full-duplex PCIe data transfers, through the use of write combined memory, and by adding the ability for the programmer to share pinned memory with multiple GPUs.

A.2 Host code

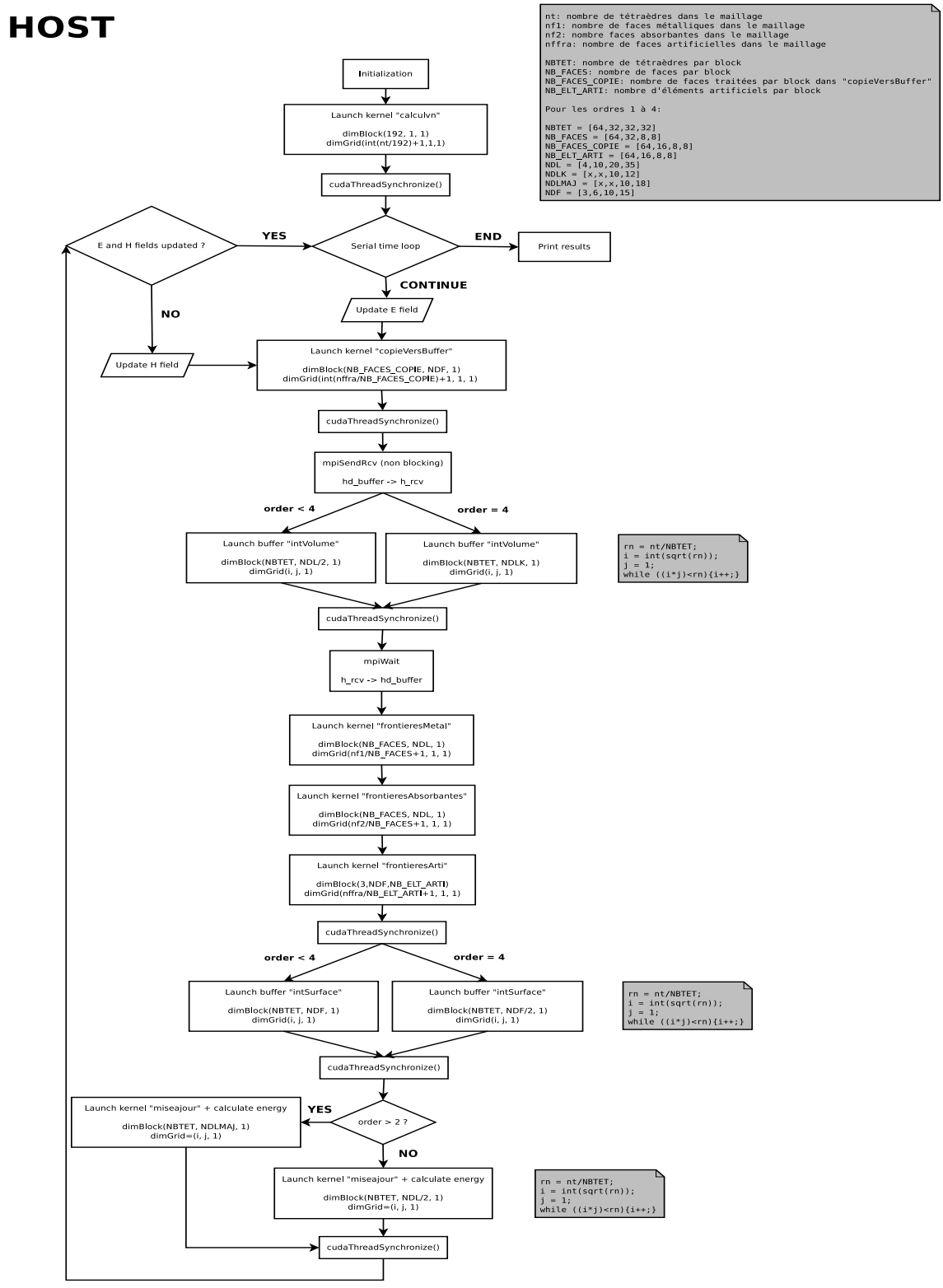


Figure A.20 – Implementation of the main serial time loop of the DGTD-method on the host

A.3 Pseudo-code CUDA

The following algorithms are written for NVidia's GT200 graphic processing unit (GPU) as well as NVIDIA new Fermi graphics cards :

- Algorithms without green lines refer to GPU implementation on NVIDIA GT200 family,
- Algorithms with green lines correspond to GPU implementation on NVIDIA Fermi family.

$name_S \rightarrow$ data in shared memory
$name_G \rightarrow$ data in global memory
$name_C \rightarrow$ data in constant memory

Algorithm 1 $intVolume\_order2$

Require: $blockDim(NBTET, NDL)$ or $blockDim(NBTET, NDL/2)$
Require: $buffer_S$ of size $3 \times NBTET \times NDL$
Require: $normalVector_S$ of size $3 \times 3 \times NBTET$ ($vn2$, $vn3$, $vn4$ of size $3 \times NBTET$)

1: $tetra = threadIdx.x$
2: $k = threadIdx.y$
3: $k2 = threadIdx.y + NDL/2$
4: $id\_block = blockIdx.x + blockIdx.y * gridDim.x$
5: $tetra\_tot = tetra + blockDim.x * id\_block$
6: $normalVector_S \leftarrow normalVector_G$
7: $buffer_S \leftarrow Field_G$ {load from texture}
8: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
9: $fl = (0.0f, 0.0f, 0.0f)$
10: $fl2 = (0.0f, 0.0f, 0.0f)$
11: **for** $i = 0$ to $NDL - 1$ **do**
12: $df_x = rdx[k][i]_C \times (vn2_x + vn3_x + vn4_x)$
13: $df_y = rdx[k][i]_C \times (vn2_y + vn3_y + vn4_y)$
14: $df_z = rdx[k][i]_C \times (vn2_z + vn3_z + vn4_z)$
15: $\vec{fl} = \vec{buffer_S} \wedge \vec{df}$
16: $df_x = rdx[k2][i]_C \times (vn2_x + vn3_x + vn4_x)$
17: $df_y = rdx[k2][i]_C \times (vn2_y + vn3_y + vn4_y)$
18: $df_z = rdx[k2][i]_C \times (vn2_z + vn3_z + vn4_z)$
19: $\vec{fl2} = \vec{buffer_S} \wedge \vec{df}$
20: **end for**
21: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
22: $\vec{buffer_S}[tetra][k] = \vec{fl}$
23: $\vec{buffer_S}[tetra][k2] = \vec{fl2}$
24: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
25: $flux_G \leftarrow buffer_S$ {write in a coalesced way}

Algorithm 2 $intSurface\_order2$

Require: $blockDim(NBTET, NDF)$ or $blockDim(NBTET, NDF/2)$
Require: $float\ \ surfaceField_S$ of size $3 \times NBTET \times NDF$
Require: $float3\ \ flux_S$ of size $3 \times NBTET \times NDL$

1: $tetra = threadIdx.x$
2: $k = threadIdx.y$
3: $k2 = threadIdx.y + NDL/2$
4: $id\_block = blockIdx.x + blockIdx.y * gridDim.x$
5: $tetra\_tot = tetra + blockDim.x * id\_block$
6: $flux_S \leftarrow flux_G$ {load in a coalesced way}
7: **for** $face = 0$ to 3 **do**
8: $neighborElement = neigborElement_G[tetra\_tot + nt_c \times face]$
9: $int2\ \ permutationElement = permutationElement_G[tetra\_tot + nt_c \times face]$
10: $int2\ \ dofElement1 = ipm_C[k][permutationElement]$
11: $index = 3 * NDL * tetra\_tot + 3 * dofElement1.x$
12: $surfaceField_S[tetra][k] \leftarrow field_G[index]$ {load from texture}
13: $index = 3 * NDL * neighborElement + 3 * dofElement1.y$
14: $surfaceField_S[tetra][k] \leftarrow +field_G[index]$ {load from texture}
15: $int2\ \ dofElement2 = ipm_C[k2][permutationElement]$
16: $index = 3 * NDL * tetra\_tot + 3 * dofElement2.x$
17: $surfaceField_S[tetra][k2] = field_G[index]$
18: $index = 3 \times NDL \times neighborElement + 3 * dofElement2.y$
19: $surfaceField_S[tetra][k2] = +field_G[index]$
20: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
21: $fl = (0.0f, 0.0f, 0.0f)$
22: **for** $i = 0$ to $NDF - 1$ **do**
23: $\vec{fl} += surfaceField[tetra][i] \times surfaceMatrice_c[k][i]$
24: **end for**
25: $flux_s[tetra][\vec{dof}Element.x] += 0.5f \times (\vec{normal}Face \wedge \vec{fl})$
26: $fl = (0.0f, 0.0f, 0.0f)$
27: **for** $i = 0$ to $NDF - 1$ **do**
28: $\vec{fl} += surfaceField[tetra][i] \times surfaceMatrice_c[k2][i]$
29: **end for**
30: $flux_s[tetra][\vec{dof}Element.y] += 0.5f \times (\vec{normal}Face \wedge \vec{fl})$
31: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
32: **end for**
33: $flux_G \leftarrow flux_S$ {write in a coalesced way}

Algorithm 3 *updateField_E2*

Require: $blockDim(NBTET, NDL)$ or $blockDim(NBTET, NDL/2)$
Require: $buffer_S$ of size $3 \times NBTET \times NDL$

1: $tetra = threadIdx.x$
2: $k = threadIdx.y$
3: $k2 = threadIdx.y + NDL/2$
4: $id\_block = blockIdx.x + blockIdx.y * gridDim.x$
5: $tetra\_tot = tetra + blockDim.x * id\_block$
6: $buffer_S \leftarrow flux_G$ {load in a coalesced way}
7: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
8: $fl = (0.0f, 0.0f, 0.0f)$
9: $fl2 = (0.0f, 0.0f, 0.0f)$
10: **for** $i = 0$ to $NDL - 1$ **do**
11: $\quad \vec{fl} = \vec{buffer_S} \times amat_C[k][i]$
12: $\quad \vec{fl2} = \vec{buffer_S} \times amat_C[k2][i]$
13: **end for**
14: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
15: $buffer_S \leftarrow fieldE_G$ {load in a coalesced way}
16: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
17: $den = 2.0f \times epsilon_G[tetra\_tot] + dt \times sigma_G[tetra\_tot]$
18: $dte = (2.0f \times epsilon_G[tetra\_tot] - dt \times sigma_G[tetra\_tot])/den$
19: $dtf = 2.0f \times dt/(volumeElement_G[tetra\_tot] \times den$
20: $\vec{buffer_S}[k][tetra] = dte \times \vec{buffer_s}[k][tetra] + dtf * \vec{fl}$
21: $\vec{buffer_S}[k2][tetra] = dte \times \vec{buffer_s}[k2][tetra] + dtf * \vec{fl}$
22: $buffer_S[k][tetra].z- = dt \times gaussianSource[tetra\_tot][k]$
23: $buffer_S[k2][tetra].z- = dt \times gaussianSource[tetra\_tot][k2]$
24: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
25: $fieldE_G \leftarrow buffer_S$ {write in a coalesced way}

Algorithm 4 *updateField_H2*

Require: $blockDim(NBTET, NDL)$ or $blockDim(NBTET, NDL/2)$
Require: $buffer_S$ of size $3 \times NBTET \times NDL$
Require: $elementVolume_S$ of size $NBTET$

1: $tetra = threadIdx.x$
2: $k = threadIdx.y$
3: $k2 = threadIdx.y + NDL/2$
4: $id\_block = blockIdx.x + blockIdx.y * gridDim.x$
5: $tetra\_tot = tetra + blockDim.x * id\_block$
6: $elementVolume_S \leftarrow elementVolume_G$ {load in a coalesced way}
7: $buffer_S \leftarrow flux_G$ {load in a coalesced way}
8: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
9: $fl = (0.0f, 0.0f, 0.0f)$
10: $fl2 = (0.0f, 0.0f, 0.0f)$
11: **for** $i = 0$ to $NDL - 1$ **do**
12: $\quad \vec{fl} = \vec{buffer_S} \times amat_C[k][i]$
13: $\quad \vec{fl2} = \vec{buffer_S} \times amat_C[k2][i]$
14: **end for**
15: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
16: $buffer_S \leftarrow fieldH_G$ {load in a coalesced way}
17: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
18: $\vec{buffer_S}[k][tetra] -= (dt \times \vec{fl})/elementVolume_S[tetra]$
19: $\vec{buffer_S}[k2][tetra] -= (dt \times \vec{fl2})/elementVolume_S[tetra]$
20: $\_\_\_\_\_MemoryFence\_\_\_\_\_$
21: $fieldH_G \leftarrow buffer_S$ {write in a coalesced way}

Bibliographie

[Abramowitz 1964] M. Abramowitz et I. Stegun. Handbook of mathematical functions with formulas, graphs, and mathematical tables. Dover Publications, 1964. (Cité en page 40.)

[Ainsworth 2003] M. Ainsworth et J. Coyle. *Hierarchic finite element bases on unstructured tetrahedral meshes*. Int. J. Numer. Meth. Engng., vol. 58, pages 2103–2130, 2003. (Cité en page 104.)

[Appell 1926] P. Appell et J. Kampé de Fériet. *Fonctions hypergéométriques et hypersphériques. Polynômes d'Hermite*. Paris, gauthier-villars édition, 1926. (Cité en page 35.)

[Baldassari 2009] C. Baldassari. *Modélisation et simulation numérique pour la migration terrestre par équation d'ondes*. PhD thesis, Université de Pau et des Pays de l'Adour, Décembre 2009. (Cité en page 121.)

[Bécache 2005] E. Bécache, P. Joly et J. Rodriguez. *Space-time mesh refinement for elastodynamics. Numerical results*. Comput. Methods Appl. Mech. Engrg., vol. 194, pages 355–366, 2005. (Cité en page 120.)

[Bernacki 2006a] M. Bernacki, L. Fezoui, S. Lanteri et S. Piperno. *Parallel Discontinuos Galerkin unstructured mesh solvers for the calculation of three-dimensional wave propagation problems*. Appl. Math. Model., vol. 30, no. 8, pages 744–763, 2006. (Cité en page 155.)

[Bernacki 2006b] M. Bernacki, S. Lanteri et S. Piperno. *Time-domain parallel simulation of heterogeneous wave propagation on unstructured grids using explicit, non-diffusive, discontinuous Galerkin methods*. J. Comput. Acoust., vol. 14, no. 1, pages 57–81, 2006. (Cité en page 155.)

[Bernardi 2000] P. Bernardi, M. Cavagnaro, S. Pisa et E. Piuzzi. *Specific absorption rate and temperature increases in the head of a cellular phone user*. IEEE Trans. Microw. Theory Tech., vol. 48, no. 7, pages 1118–1126, 2000. (Cité en page 3.)

[Bernardi 2001] P. Bernardi, M. Cavagnaro, S. Pisa et E. Piuzzi. *Power absorbtion and temperature elevation induced in the human head by a dual-band monopole-helix antenna phone*. IEEE Trans. Microw. Theory Tech., vol. 49, no. 12, pages 2539–2546, 2001. (Cité en pages 3 et 162.)

[Boissonnat 2005] J.-D. Boissonnat et S. Oudot. *Provably good sampling and meshing of surfaces*. Graphical Models, vol. 67, no. 5, pages 405–451, 2005. (Cité en page 162.)

[Bondeson 2005] A. Bondeson, T. Rylander et P. Ingelström. Computational electromagnetics, texts in Applied Mathematics, volume 51. Springer, 1st ed. édition, 2005. (Cité en page 6.)

[Bonnet 1997] P. Bonnet et X. Ferrieres. *Numerical modeling of scattering problems using a time domain finite volume method*. J. Electromagn. Waves Appl., vol. 11, pages 1165–1189, 1997. (Cité en page 119.)

[Bossavit 1988] A. Bossavit. *A rationale for edge-elements in 3-D fields computations*. IEEE Trans. Magn., vol. 24, no. 1, pages 74–79, 1988. (Cité en page 6.)

[Bossavit 1990] A. Bossavit. *Solving Maxwell equations in a closed cavity, and the question of spurious modes*. IEEE Trans. Magn., vol. 26, no. 2, pages 702–705, 1990. (Cité en page 6.)

[Buffa 2007] A. Buffa, P. Houston et I. Perugia. *Discontinuous Galerkin computation of the Maxwell eigenvalues on simplicial meshes*. J. Comput. Appl. Math., vol. 204, no. 2, pages 317–333, 2007. (Cité en page 21.)

[Cabel 2011] T. Cabel, J. Charles et S. Lanteri. *Multi-GPU acceleration of a DGTD method for modelling human exposure to electromagnetic waves*. Rapport technique, INRIA Technical Report RT-7592, 2011. (Cité en page 155.)

[Camart 1996] J-C. Camart, D. Despretz, M. Chivé et J. Pribetich. *Modeling of various kinds of applicators used for microwave hyperthermia based on the FDTD method*. IEEE Trans. Microw. Theory Tech., vol. 44, no. 10, pages 1811–1818, 1996. (Cité en page 162.)

[Cangellaris 1991] A.C. Cangellaris et D.B. Wright. *Analysis of the numerical error caused by the stair-stepped approximation of a conductiong boundary in FDTD simulations of electromagnetic phenomena*. IEEE Trans. Antennas and Propag., vol. 39, no. 10, pages 1518–1525, 1991. (Cité en page 119.)

[Canouet 2005] N. Canouet, L. Fezoui et S. Piperno. *Discontinuous Galerkin time-domain solution of Maxwell's equations on locally-refined nonconforming cartesian grids*. COMPEL, vol. 24, no. 4, pages 1381–1401, 2005. (Cité en page 21.)

[Catella 2010] A. Catella, V. Dolean et S. Lanteri. *An implicit discontinuous Galerkin time-domain method for two-dimensional electromagnetic wave propagation*. COMPEL, vol. 29, no. 3, pages 602–625, 2010. (Cité en page 123.)

[Chen 2005] M.-H. Chen, B. Cockburn et F. Reitich. *High-order RKDG methods for computational electromagnetics*. J. Sci. Comput., vol. 22, no. 1, pages 205–226, 2005. (Cité en page 21.)

[Chevalier 1997] M.W. Chevalier et R.J. Luebbers. *FDTD local grid with material transverse*. IEEE Trans. Antennas and Propag., vol. 45, no. 3, pages 411–421, 1997. (Cité en page 120.)

[Chew 1993] L.P. Chew. *Guaranteed-quality mesh generation for curved surfaces*. In 9th Annual ACM Symposium Computational Geometry, pages 274–280. ACM Press, 1993. (Cité en page 162.)

[Cioni 1993] J.-P. Cioni, L. Fezoui et H. Steve. *A parallel time-domain Maxwell solver using upwind schemes and triangular meshes*. IMPACT Comput. Sci. Eng., vol. 5, no. 3, pages 215–247, 1993. (Cité en page 8.)

[Clatz 2000] O. Clatz, S. Lanteri, S. Oudot, J.-P. Pons, S. Piperno, G. Scarella et J. Wiart. *Modélisation numérique réaliste de l'exposition des tissus de la tête à un champ électromagnétique issu d'un téléphone mobile*. In 13ème Colloque International et Exposition sur la Compatibilité Electromagnétique (Saint Malo, France), pages 377–397, 2000. (Cité en page 3.)

[Cockburn 1989] B. Cockburn et C.-W. Shu. *TVB Runge-Kutta local projection discontinuous Galerkin method for conservation laws II : general framework*. Math. Comput., vol. 52, no. 186, pages 411–435, 1989. (Cité en page 21.)

[Cockburn 2004] B. Cockburn, F. Li et C.-W. Shu. *Locally divergence-free discontinuous Galerkin methods for the Maxwell equations*. J. Comput. Phys., vol. 194, no. 1, pages 588–610, 2004. (Cité en page 21.)

[Cohen 1998] G. Cohen et P. Monk. *Gauss point mass lumping schemes for Maxwell's equations*. Numer. Meth. Part. Diff. Eqns., vol. 14, pages 63–68, 1998. (Cité en page 7.)

[Cohen 2006] G. Cohen, X. Ferrieres et S. Pernet. *A spatial high-order hexahedral discontinuous Galerkin method to solve Maxwell's equations in time domain*. J. Comput. Phys., vol. 217, pages 340–363, 2006. (Cité en pages 21, 120 et 154.)

[Coifman 1993] R. Coifman, V. Rokhlin et S. Wandzura. *The fast multipole method for the wave equation : a pedestrian approach*. IEEE Trans. Antennas Propag., vol. 35, no. 7, pages 7–12, 1993. (Cité en page 6.)

[Collino 2003a] F. Collino, T. Fouquet et P. Joly. *A conservative space-time mesh refinement method for the 1-D wave equation. Part I : Construction*. Numer. Math., vol. 95, pages 197–221, 2003. (Cité en page 120.)

[Collino 2003b] F. Collino, T. Fouquet et P. Joly. *A conservative space-time mesh refinement method for the 1-D wave equation. Part II : Analysis*. Numer. Math., vol. 95, pages 223–251, 2003. (Cité en page 120.)

[Collino 2006] F. Collino, T. Fouquet et P. Joly. *Conservative space-time mesh refinement methods for the FDTD solution of Maxwell's equations*. J. Comput. Phys., vol. 211, no. 1, pages 9–35, 2006. (Cité en page 120.)

[Conil 2008] E. Conil, A. Hadjem, F. Lacroux, M.F. Wong et J. Wiart. *Variability analysis of SAR from 20 MHz to 2.4 GHz for different adult and child models using finite-difference time-domain*. Phys. Med. Biol., vol. 53, pages 151–1525, 2008. (Cité en page 162.)

[Demkowicz 2005] L. Demkowicz et D. Xue. *Control in geometry induced in hp finite element simulations*. Numer. Anal. and Model., vol. 2, no. 3, pages 283–300, 2005. (Cité en page 7.)

[Diaz 2009] J. Diaz et M.J. Grote. *Energy conserving explicit local time-stepping for second-order wave equations*. J. Sci. Comput., vol. 31, pages 1985–2014, 2009. (Cité en pages 9, 121, 124, 126, 132, 133 et 179.)

[Dodson 1997] S. Dodson, S.P. Walker et M.J. Bluck. *Implicitness and stability of time domain integral equation scattering analysis*. Appl. Comput. Electromagn. Soc. J., vol. 13, no. 3, pages 291–301, 1997. (Cité en page 6.)

[Dolean 2008] V. Dolean, H. Fol, S. Lanteri et R. Perrussel. *Solution of the time-harmonic Maxwell equations using discontinuous Galerkin methods*. J. Comput. Appl. Math., vol. 218, no. 2, pages 435–445, 2008. (Cité en page 21.)

[Dolean 2010] V. Dolean, H. Fahs, L. Fezoui et S. Lanteri. *Locally implicit discontinuous Galerkin method for time domain electromagnetics*. J. Comput. Phys., vol. 229, no. 2, pages 512–526, 2010. (Cité en page 120.)

[Dubiner 1991] M. Dubiner. *Spectral methods on triangles and other domains*. J. Sci. Comput., vol. 6, no. 4, pages 345–390, 1991. (Cité en pages 35 et 105.)

[Dumbser 2007] M. Dumbser, M. Käser et E.F. Toro. *An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes - V. Local time stepping and p-adaptivity*. Geophys. J. Int., vol. 171, pages 695–717, 2007. (Cité en page 121.)

[Dunn 1996] D. Dunn, C.M. Rappaport et A.J. Terzuoli. *FDTD verification of deep-set brain tumor hyperthermia using a spherical microwave source distribution*. IEEE Trans. Microw. Theory Tech., vol. 44, pages 1769–1777, 1996. (Cité en page 162.)

[Edelvik 2000] F. Edelvik et G. Ledfelt. *Explicit hybrid time domain solver for the Maxwell equations in 3D*. J. Sci. Comput., vol. 15, no. 1, pages 61–78, 2000. (Cité en page 119.)

[Ergin 1998] A.A. Ergin, B. Shanker et E. Michielssen. *Fast evaluation of three-dimensional transient wave fields using diagonal translation operators*. J. Comput. Phys., vol. 146, no. 1, pages 157–180, 1998. (Cité en page 6.)

[Ezziani 2010] A. Ezziani et P. Joly. *Local time stepping and discontinuous Galerkin methods for symmetric first order hyperbolic systems*. J. Comput. Appl. Math., vol. 234, pages 1886–1895, 2010. (Cité en page 121.)

[Fahs 2008] H. Fahs. *Méthodes de type Galerkin discontinu d'ordre élevé pour la résolution numérique des équations de Maxwell instationnaires sur des maillages simplexes non-conformes*. PhD thesis, Ecole Doctorale Sciences Fondamentales et Appliquées, 2008. (Cité en pages 8 et 35.)

[Fahs 2009] H. Fahs. *Development of a hp-like discontinuous Galerkin time-domain method on non-conforming simplicial meshes for electromagnetic wave propagation*. Int. J. Numer. Anal. Mod., vol. 6, pages 193–216, 2009. (Cité en pages 120 et 153.)

[Farouki 2003] R.T. Farouki, T.N.T. Goodman et T. Sauer. *Construction of orthogonal bases for polynomials in Bernstein form on triangular and simplex domains*. Computer Aided Geometric Design, vol. 20, no. 4, pages 209–230, 2003. (Cité en page 35.)

[Fezoui 2005] L. Fezoui, S. Lanteri, S. Lohrengel et S. Piperno. *Convergence and stability of a discontinuous Galerkin time-domain method for the 3D heterogeneous Maxwell equations on unstructured meshes*. ESAIM : Math. Model. and Numer. Anal., vol. 39, no. 6, pages 1149–1176, 2005. (Cité en pages 7, 8, 21, 22, 32, 34, 106, 116, 120, 121, 154 et 155.)

[Gandhi 2001] O.P. Gandhi, Q.-X. Li et G. Kang. *Temperature rise for the human head for cellular telephones and for peak SARs prescribed in safety guidelines*. IEEE Trans. Microw. Theory Tech., vol. 49, no. 9, pages 1607–1613, 2001. (Cité en page 162.)

[Garcia 2004] J. Rodriguez Garcia. *Raffinement de Maillage Spatio-Temporel pour les équations de l'élastodynamique*. PhD thesis, Université Paris Dauphine, Décembre 2004. (Cité en page 120.)

[George 1991] P.-L. George, F. Hecht et E. Saltel. *Automatic mesh generator with specified boundary*. Comput. Methods Appl. Mech. Engrg., vol. 92, pages 269–288, 1991. (Cité en page 163.)

[Gödel 2009] N. Gödel, S. Schomann, T. Warburton et M. Clemens. *Local timestepping discontinuous Galerkin methods for electromagnetic RF field problems*. In 3rd European Conference on Antennas and Propagation (EuCAP 2009), pages 2149–2153, 2009. (Cité en page 121.)

[Gödel 2010] N. Gödel, N. Nunn, T. Warburton et M. Clemens. *Scalability of higher-order discontinuous Galerkin FEM computations for solving electromagnetic wave propagation problems on GPU clusters*. IEEE Trans. Magn., vol. 46, no. 8, pages 3469–3472, 2010. (Cité en page 154.)

[Grote 2009] M.J. Grote et T. Mitkova. *Explicit local time-stepping methods for Maxwell's equations*. Rapport technique 2009-02, University of Basel, Department of Mathematics, 2009. (Cité en pages 9, 121, 124, 126 et 179.)

[Hesthaven 2002] J.S. Hesthaven et T. Warburton. *Nodal high-order methods on unstructured grids. I. Time-domain solution of Maxwell's equations*. J. Comput. Phys., vol. 181, pages 186–221, 2002. (Cité en pages 21, 120 et 154.)

[Hesthaven 2004a] J.S. Hesthaven et T. Warburton. *High-order accurate methods for time-domain electromagnetics*. Comp. Mod. Engin. Sci., vol. 5, no. 5, pages 395–408, 2004. (Cité en page 21.)

[Hesthaven 2004b] J.S. Hesthaven et T. Warburton. *High-order nodal discontinuous Galerkin methods for the Maxwell's eigenvalue problem*. Royal Soc. London Ser. A, vol. 362, pages 493–524, 2004. (Cité en page 21.)

[Hiptmair 2001] R. Hiptmair. *Higher order Whitney forms*. Prog. In Electr. Res. (PIER), vol. 32, pages 271–299, 2001. (Cité en page 7.)

[Houston 2003] P. Houston, I. Perugia et D. Schötzau. *hp-DGFEM for Maxwell's equations*. In Numerical Mathematics and Advanced Applications (ENUMATH 2001) (F. Brezzi, A. Buffa, S. Corsaro, and A. Murli, eds), pages 785–794. Springer-Verlag, 2003. (Cité en page 21.)

[Houston 2004] P. Houston, I. Perugia et D. Schötzau. *Mixed discontinuous Galerkin approximation of the Maxwell operator*. SIAM J. Numer. Anal., vol. 42, no. 1, pages 434–459, 2004. (Cité en page 21.)

[Houston 2005] P. Houston, I. Perugia et D. Schötzau. *Mixed discontinuous Galerkin approximation of the Maxwell operator : non-stabilized formulation*. J. Sci. Comput., vol. 22, pages 325–356, 2005. (Cité en page 21.)

[Ji 2005] X. Ji, T. Lu, W. Cai et P. Zhang. *Discontinuous Galerkin time domain (DGTD) methods for the study of 2-D waveguide-coupled microring resonators*. Journal of Lightwave Technology, vol. 23, no. 11, pages 3864–3874, 2005. (Cité en page 21.)

[Joly 2005] P. Joly et J. Rodriguez. *An error analysis of conservative space-time mesh refinement methods for the one-dimensional wave equation*. SIAM J. Numer. Anal., vol. 43, pages 825–859, 2005. (Cité en page 120.)

[Jung 2003] B.H. Jung, Y.S. Chung et T.K. Sarkar. *Time-domain EFIE, MFIE, and CFIE formulations using Laguerre polynomials as temporal basis functions for analysis of transient scattering from arbitrarily shaped conducting structures*. Progress in Electromagnetics Research, vol. 39, pages 1–45, 2003. (Cité en page 7.)

[Jung 2007] B.H. Jung, Z. Li, T.K. Sarkar, M. Salazar-Palma et M. Yuan. *A comparison of marching-on-in-time method with marching-on-in-degree method for the TDIE solver*. Progress in Electromagnetics Research, vol. 70, pages 281–296, 2007. (Cité en page 7.)

[Kabakian 2004] A.V. Kabakian, V. Shankar et W.F. Hall. *Unstructured grid-based discontinuous Galerkin method for broadband electromagnetic simulations*. J. Sci. Comput., vol. 20, no. 3, pages 405–431, 2004. (Cité en page 21.)

[Karniadakis 2005] G. Em Karniadakis et S.J. Sherwin. Spectral/*hp* element methods for computational fluid dynamics. Numerical Mathematics and Scientific Computation. Oxford University Press, 2005. (Cité en pages 40 et 52.)

[Kim 2004] J. Kim et Y. Rahmat-Samii. *Implanted antennas inside a human body : simulations, designs, and characterizations*. IEEE Trans. Microw. Theory Tech., vol. 52, no. 8, pages 1934–1943, 2004. (Cité en page 162.)

[Kirk 2010] D.B. Kirk et W.W. Hwu. Programming Massively Parallel Processors : A Hands-on Approach. Elsevier, 2010. (Cité en page 185.)

[Klöckner 2009] A. Klöckner, T. Warburton, J. Bridge et J.S. Hesthaven. *Nodal discontinuous Galerkin methods on graphic processors*. J. Comput. Phys., vol. 228, pages 7863–7882, 2009. (Cité en page 154.)

[Komatitsch 2009] D. Komatitsch, D. Michéa et G. Erlebacher. *Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA*. J. Parallel Distrib. Comput., vol. 69, pages 451–460, 2009. (Cité en page 154.)

[Komatitsch 2010a] D. Komatitsch, G. Erlebacher, D. Göddeke et D. Michéa. *High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster*. J. Comput. Phys., vol. 229, no. 20, pages 7692–7714, 2010. (Cité en pages 154 et 155.)

[Komatitsch 2010b] D. Komatitsch, D. Göddeke, G. Erlebacher et D. Michéa. *Modeling the propagation of elastic waves using spectral elements on a cluster of 192 GPUs*. Comput. Sci. Res. Dev., vol. 25, pages 75–82, 2010. (Cité en page 154.)

[Koorwinder 1975] T. Koorwinder. *Two-variable analogues of the classical orthogonal polynomials*. In Theory and Application of Special Functions, pages 435–495. R. A. Askey ed., Academic Press, 1975. (Cité en page 35.)

[Kopriva 2000] D. Kopriva, S.L. Woodruff et M.Y. Hussaini. *Discontinuous spectral element approximation of Maxwell's equations*. In Discontinuous Galerkin Methods : Theory, Computation and Applications (B. Cockburn, G.E. Karniadakis and C.W. Shu, eds), volume 11, pages 355–362. Springer-Verlag, 2000. (Cité en page 21.)

[Kunz 1981] K.S. Kunz et L. Simpson. *A technique for increasing the resolution of finite-difference solutions to the Maxwell equations*. IEEE Trans. Electromagn. Compat., vol. EMC-23, pages 419–422, 1981. (Cité en page 120.)

[Lin 2000] J.C. Lin, S. Hirai, C.-L. Chiang, W.-L. Hsu, J.-L. Su et Y.-J. Wang. *Computer simulation and experimental studies of SAR distributions of interstitial arrays of sleeved-slot microwave antennas for hyperthermia treatment of brain tumors*. IEEE Trans. Microw. Theory Tech., vol. 48, no. 11, pages 2191–2198, 2000. (Cité en pages 3 et 162.)

[Liu 2010] L. Liu, X. Li et F.Q. Hu. *Non uniform time-step Runge-Kutta discontinuous Galerkin method for Computational Aeroacoustics*. J. Comput. Phys., vol. 229, pages 6874–6897, 2010. (Cité en page 121.)

[Min 2005] M. Min. *Discontinuous Galerkin method based on quadrilateral mesh for Maxwell's equations*. In Proc. of IEEE/ACES on Wireless Communications and Applied Computational Electromagnetics, pages 724–727, 2005. (Cité en page 21.)

[Monk 2003] P. Monk. Finite element methods for Maxwell's equations. Numerical Mathematics and Scientific Computation. Oxford University Press, 2003. (Cité en page 7.)

[Montseny 2008] E. Montseny, S. Pernet, X. Ferriéres et G. Cohen. *Dissipative terms and local time-stepping improvements in a spatial high order discontinuous Galerkin scheme for the time-domain Maxwell's equations*. J. Comput. Phys., vol. 227, no. 14, pages 6795–6820, 2008. (Cité en page 121.)

[Nedelec 1980] J.C. Nedelec. *Mixed finite elements in* $\mathbb{R}^3$. Numer. Math., vol. 35, pages 315–341, 1980. (Cité en pages 6 et 7.)

[Nedelec 1986] J.C. Nedelec. *A new family of mixed finite elements in* $\mathbb{R}^3$. Numer. Math., vol. 50, pages 57–81, 1986. (Cité en pages 6 et 7.)

[NVIDIA 2011] NVIDIA. *CUDA C programming guide version 4.0*. NVIDIA Corporation documentation, 2011. (Cité en pages 170 et 181.)

[Owens 1998] R.G. Owens. *Spectral approximations on the triangle*. Proc. R. Soc. Lond. A, vol. 454, no. 1971, pages 857–872, 1998. (Cité en page 35.)

[Paulsen 1991] K.D. Paulsen et D.R. Lynch. *Elimination of vector parasites in finite element Maxwell solutions*. IEEE Trans. Microw. Theory Tech., vol. 39, pages 395–404, 1991. (Cité en page 6.)

[Piperno 2000] S. Piperno. *L^2-stability of the upwind first order finite volume scheme for the Maxwell equations in two and three dimensions on arbitrary unstructured meshes*. RAIRO : Model. Math. Anal. Numer., vol. 34, pages 139–158, 2000. (Cité en page 7.)

[Piperno 2002] S. Piperno, M. Remaki et L. Fezoui. *A nondiffusive finite volume scheme for the three-dimensional Maxwell's equations on unstructured meshes*. SIAM J. Numer. Anal., vol. 39, no. 6, pages 2089–2108, 2002. (Cité en pages 8, 119 et 120.)

[Piperno 2003] S. Piperno et L. Fezoui. *A centered discontinuous Galerkin finite volume scheme for the 3D heterogeneous Maxwell equations on unstructured meshes*. Rapport technique, INRIA Technical Report RT-4733, 2003. (Cité en pages 7, 8 et 120.)

[Piperno 2006a] S. Piperno. *DGTD methods using modal basis functions and symplectic local time-stepping : application to wave propagation problems*. European Journal of Computational Mechanics, vol. 15, no. 6, pages 643–670, 2006. (Cité en page 120.)

[Piperno 2006b] S. Piperno. *Symplectic local time-stepping in non-dissipative DGTD methods applied to wave propagation problems*. ESAIM : Math. Model. Numer. Anal., vol. 40, no. 5, pages 815–841, 2006. (Cité en page 120.)

[Proriol 1957] J. Proriol. *Sur une famille de polynômes à deux variables orthogonaux dans un triangle*. C. R. Acad. Sci. Paris, pages 2459–2461, 1957. (Cité en page 35.)

[Qui 2005] J.X. Qui, M. Dumbser et C.-W. Shu. *The discontinuous Galerkin method with Lax-Wendroff type time discretizations*. Comput. Methods Appl. Mech. Eng., vol. 194, pages 4528–4543, 2005. (Cité en page 121.)

[Ratiu 2003] P. Ratiu, B. Hillen, J. Glaser et D.P. Jenkins. Medicine Meets Virtual Reality 11 - NextMed : Health Horizon, volume 11, chapitre Visible Human 2.0 - the next generation, pages 275–281. IOS Press, 2003. (Cité en page 162.)

[Reed 1973] W.H. Reed et T.R. Hill. *Triangular mesh methods for the neutron transport equation*. Rapport technique LA-UR-73-479, Los Alamos National Laboratory, Los Alamos, New Mexico, 1973. (Cité en pages 21 et 120.)

[Reitich 2004] F. Reitich et K.K. Tamma. *State-of-the-art, trends and directions in Computational Electromagnetics*. CMES Comput. Model. Eng. Sci., vol. 5, pages 287–294, 2004. (Cité en page 4.)

[Remacle 2001] J.-F. Remacle, K. Pinchedez, J. Flaherty et M. Shephard. *An efficient local time-stepping discontinuous Galerkin scheme for adaptive transient computations*. Computer Methods in Applied Mechanics and Engineering, 2001. submitted. (Cité en page 21.)

[Remacle 2003] J.-F. Remacle, J. Flaherty et M. Shephard. *An adaptative discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems*. SIAM Review, vol. 45, no. 1, pages 53–72, 2003. (Cité en page 21.)

[Remaki 2000] M. Remaki. *A new finite volume scheme for solving Maxwell's system*. COMPEL, vol. 19, pages 913–931, 2000. (Cité en page 7.)

[Rynne 1990] B.P. Rynne et P.D. Smith. *Stability of time marching algorithms for the electric field integral equation.* J. Electromagn. Waves Appl., vol. 12, pages 1181–1205, 1990. (Cité en page 6.)

[Sanders 2010] J. Sanders et E. Kandrot. CUDA by Example : An Introduction to General-Purpose GPU Programming. Addison-Wesley, 2010. (Cité en page 185.)

[Schomann 2010] S. Schomann, N. Gödel, T. Warburton et M. Clemens. *Local timestepping techniques using Taylor expansion for modeling electromagnetic wave propagation with discontinuous Galerkin-FEM.* IEEE Trans. Magn., vol. 46, no. 8, pages 3504–3507, 2010. (Cité en page 121.)

[Sherwin 1995] S.J. Sherwin et G. Karniadakis. *A new triangular and tetrahedral basis for high-order (hp) finite element methods.* Int. J. Numer. Meth. Engng., vol. 38, no. 22, pages 3775–3803, 1995. (Cité en pages 105 et 106.)

[Siauve 2003] N. Siauve, L. Nicolas, C. Vollaire et C. Marchal. *3D modeling of electromagnetic fields in local hyperthermia.* Eur. Phys. J. AP, vol. 21, pages 243–250, 2003. (Cité en page 3.)

[Solin 2004] P. Solin, K. Segeth et I. Dolezel. Higher-order Finite Element Methods. Chapman and Hall, crc ed. édition, 2004. (Cité en pages 102 et 104.)

[Song 1995] J.M. Song et W.C. Chew. *Multilevel fast-multipole algorithm for solving combined field integral equations of electromagnetic scattering.* Microwave Opt. Technol. Lett., vol. 10, no. 1, pages 14–19, 1995. (Cité en page 6.)

[Spachmann 2002] H. Spachmann, R. Schuhmann et T. Weiland. *High order explicit time integration schemes for Maxwell's equations.* Int. J. Numer. Model., vol. 15, no. 6, pages 419–437, 2002. (Cité en page 58.)

[Sun 1995] D. Sun, J. Manges, X. Yuan et Z. Cendes. *Spurious modes in finite element methods.* IEEE Antennas Propag. Mag., vol. 37, pages 12–24, 1995. (Cité en page 6.)

[Taflove 2005] A. Taflove et S.C. Hagness. Computational Electrodynamics : The Finite-Difference Time-Domain Method. Norwood, MA : Artech House, 3rd ed. édition, 2005. (Cité en page 119.)

[Taube 2009] A. Taube, M. Dumbser, C.D. Munz et R. Schneider. *A high-order discontinuous Galerkin method with time-accurate local time stepping for the Maxwell equations.* Int. J. Numer. Model., vol. 22, pages 77–103, 2009. (Cité en page 121.)

[Wang 1991] J.J.H. Wang. Generalized Moment methods in Electromagnetics, Formulation and Computer Solution of Integral Equations. John Wiley and Sons Inc, 1991. (Cité en page 5.)

[Warburton 2000] T. Warburton. *Application of the discontinuous Galerkin method to Maxwell's equations using unstructured polymorphic hp-finite elements.* In Discontinuous Galerkin Methods : Theory, Computation and Applications (B. Cockburn, G.E. Karniadakis and C.W. Shu, eds), volume 11, pages 451–458. Springer-Verlag, 2000. (Cité en page 21.)

[Xin 2006] J. Xin et J.E. Flaherty. *Viscous stabilization of discontinuous Galerkin solutions of hyperbolic conservation laws.* Appl. Numer. Math., vol. 56, no. 3-4, pages 444–458, 2006. (Cité en page 21.)

[Xin 2012] J. Xin et W. Cai. *Well-conditioned orthonormal hierarchical* $\mathbb{L}^2$ *bases on* $\mathbb{R}^n$ *simplicial elements.* J. Sci. Comput., vol. 50, no. 2, pages 446–461, 2012. (Cité en page 179.)

[Xu 2001] Y. Xu. *Orthogonal polynomials and cubature formulae on balls, simplices and spheres.* J. Comput. Appl. Math., vol. 127, no. 1-2, pages 349–368, 2001. (Cité en page 35.)

[Yee 1966] K.S. Yee. *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media.* IEEE Trans. Antennas and Propag., vol. 14, no. 3, pages 302–307, 1966. (Cité en pages 7, 58, 119 et 162.)

Performances improvement of high-order discontinuous Galerkin methods for the numerical solution of unsteady Maxwell's equations on simplicial meshes

Abstract :
This work is concerned with the development of a flexible and efficient arbitrary high-order Discontinuous Galerkin Time Domain (DGTD) method for solving time-domain Maxwell's equations on unstructured simplicial meshes, relying on explicit time integration schemes. Electromagnetic field components are approximated locally by polynomial interpolation methods and continuity between neighbouring elements is weakly enforced by a centered scheme for the calculation of the numerical flux across mesh interfaces. The aim of this PhD thesis is to fulfill two complementary objectives. On one hand, to improve the polynomial approximation flexibility in view of the development of p-adaptive DGTD methods by studying various polynomial interpolation methods. Several aspects such as the modal or nodal nature of the associated set of basis functions, their possible hierarchical structure, the conditioning of the elementary matrices to be inverted, the spectral properties of the interpolation or the programming simplicity are investigated. On the other hand, to increase the efficiency of the temporal approximation on locally refined meshes by using a local time stepping strategy. We finally develop in this work a high performance computing methodology to exploit the inherent locality and parallelism of DGTD methods combined with the GPU computing capabilities. The combination of these brand features result in worth improvement of efficiency and in significant reduction of the computational time.

Amélioration des performances de méthodes Galerkin discontinues d'ordre élevé pour la résolution numérique des équations de Maxwell instationnaires sur des maillages simplexes

Résumé :
Cette étude concerne le développement d'une méthode Galerkin discontinue d'ordre élevé en domaine temporel (DGTD), flexible et efficace, pour la résolution des équations de Maxwell instationnaires sur des maillages simplexes destructurés et reposant sur des schémas d'intégration en temps explicites. Les composantes du champ électromagnétique sont approximées localement par des méthodes d'interpolation polynomiale et la continuité entre éléments adjacents est renforcée de façon faible par un schéma centré pour le calcul du flux numérique à travers les interfaces du maillage. L'objectif de cette thèse est de remplir deux objectifs complémentaires. D'une part, améliorer la flexibilité de l'approximation polynomiale en vue du développement de méthodes DGTD p-adaptatives par l'étude de différentes méthodes d'interpolation polynomiale. Plusieurs aspects tels que la nature nodale ou modale de l'ensemble des fonctions de bases associées, leur éventuelle structure hiérarchique, le conditionnement des matrices élémentaires à inverser, les propriétés spectrales de l'interpolation ou la simplicité de programmation sont étudiés. D'autre part, augmenter l'efficacité de l'approximation temporelle sur des maillages localement raffinés en utilisant une stratégie de pas de temps local. Nous développerons finalement dans cette étude une méthodologie de calcul haute performance pour exploiter la localité et le parallélisme inhérents aux méthodes DGTD combinés aux capacités de calcul sur carte graphique. La combinaison de ces caractéristiques modernes résulte en une amélioration importante de l'efficacité et en une réduction significative du temps de calcul.